

Artificial Intelligence with SAS®

Special Collection



Foreword by
Saurabh Gupta

The correct bibliographic citation for this manual is as follows: Gupta, Saurabh. 2018. *Artificial Intelligence with SAS®: Special Collection*. Cary, NC: SAS Institute Inc.

Artificial Intelligence with SAS® : Special Collection

Copyright © 2018, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

August 2018

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS software may be provided with certain third-party software, including but not limited to open-source software, which is licensed under its applicable third-party software license agreement. For license information about third-party software distributed with SAS software, refer to <http://support.sas.com/thirdpartylicenses>.

Table of Contents

[Navigating the Analytics Life Cycle with SAS® Visual Data Mining and Machine Learning](#)
Brett Wujek, SAS, Susan Haller, SAS, Jonathan Wexler, SAS

[Managing the Expense of Hyperparameter Autotuning](#)
Patrick Koch, SAS, Brett Wujek, SAS, Oleg Golovidov, SAS

[Analyzing Text In-Stream and at the Edge](#)
Simran Bagga, SAS

[Harvesting Unstructured Data to Reduce Anti-Money Laundering \(AML\) Compliance Risk](#)
Austin Cook, SAS, Beth Herron, SAS

[Invoiced: Using SAS® Text Analytics to Calculate Final Weighted Average Price](#)
Alexandre Carvalho, SAS

[Using SAS® Text Analytics to Assess International Human Trafficking Patterns](#)
Tom Sabo, SAS, Adam Pilz, SAS

[Biomedical Image Analytics Using SAS® Viya®](#)
Fijoy Vadakkumpadan, SAS, Saratendu Sethi, SAS

[How to Build a Recommendation Engine Using SAS® Viya®](#)
Jared Dean, SAS

Free SAS® e-Books: Special Collection

In this series, we have carefully curated a collection of papers that introduces and provides context to the various areas of analytics. Topics covered illustrate the power of SAS solutions that are available as tools for data analysis, highlighting a variety of commonly used techniques.



Discover more free SAS e-books!
support.sas.com/freesasebooks

About This Book

What Does This Collection Cover?

The broad definition of Artificial Intelligence (AI) is the simulation of human intelligence by machines. These machines can understand requests in natural (human) language, learn, observe, reason and self-correct. AI is particularly adept at processing and analyzing large amounts of data to provide targeted courses of action for human consideration. It applies machine learning, deep learning, and natural language processing (NLP) to solve actual problems. SAS embeds AI capabilities in our software to deliver more intelligent, automated solutions that help you boost productivity and unlock new possibilities.

The papers included in this special collection demonstrate how cutting-edge AI techniques can benefit your data analysis.

The following papers are excerpts from the SAS Global Users Group (SUGI) *Proceedings*. For more SUGI and SAS Global Forum *Proceedings*, visit [the online versions of the Proceedings](#).

More helpful resources are available at [support.sas.com](#) and [sas.com/books](#).

We Want to Hear from You

SAS Press books are written *by* SAS users *for* SAS users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit [sas.com/books](#) to

- Sign up to review a book
- Request information on how to become a SAS Press author
- Recommend a topic
- Provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com.

Foreword

“AI has been an integral part of SAS software for years. Today we help customers in every industry capitalize on advancements in AI, and we’ll continue embedding AI technologies like machine learning and deep learning in solutions across the SAS portfolio.” Jim Goodnight, CEO, SAS

Artificial intelligence is a topic that is on the mind of almost all of our customers at SAS. We are frequently discussing and evaluating the best way to leverage AI within an organization and how to help companies make sense of the volume and variety of data they have available and waiting to be used. Whereas in the past, this data may have taken the form of structured tabular data sets, today we are embracing opportunities with text, image and video data as well.

We are also frequently researching and developing the best ways to make artificial intelligence easier to use and easier to deploy into production. As our Chief Operating Officer, Dr. Oliver Schabenberger says, data without analytics is value not yet realized. Today, powerful AI is augmenting analytics in every area, and helping to maximize the value of the analytic tools and solutions that SAS has been championing for the last 42 years.

SAS delivers AI solutions that incorporate many different techniques, including machine learning, computer vision and natural language processing, and several ground breaking papers have been written to demonstrate these. We have carefully selected a handful from recent SAS Global Forum papers which illustrate how SAS is adding capabilities to our tools and solutions that help customers build their own AI solutions; and examples of AI solutions using our tools.

I hope you enjoy the following papers and that they further guide you down your path in building and deploying AI systems.

[Navigating the Analytics Life Cycle with SAS® Visual Data Mining and Machine Learning](#)

Brett Wujek, SAS, Susan Haller, SAS, Jonathan Wexler, SAS

Extracting knowledge from data to enable better business decisions is not a single step. It is an iterative life cycle that incorporates data ingestion and preparation, interactive exploration, application of algorithms and techniques for gaining insight and building predictive models, and deployment of models for assessing new observations. The latest release of SAS® Visual Data Mining and Machine Learning on SAS® Viya® accommodates each of these phases in a coordinated fashion with seamless transitions and common data usage. An intelligent process flow (pipeline) experience is provided to automatically chain together powerful machine learning methods for common tasks such as feature engineering, model training, ensembling, and model assessment and comparison. Ultimate flexibility is offered through incorporation of SAS® code into the pipeline, and collaboration with teammates is accomplished using reusable nodes and pipelines. This paper provides an in-depth look at all that this solution has to offer.

[Managing the Expense of Hyperparameter Autotuning](#)

Patrick Koch, SAS, Brett Wujek, SAS, Oleg Golovidov, SAS

Machine learning predictive modeling algorithms are governed by “hyperparameters” that have no clear defaults agreeable to a wide range of applications. The depth of a decision tree, number of trees in a forest or a gradient boosting tree model, number of hidden layers and neurons in each layer in a neural network, and degree of regularization to prevent overfitting are a few examples of quantities that must be prescribed. Determining the best values of machine learning algorithm hyperparameters for a specific data set can be a difficult and computationally expensive challenge. The recently released AUTOTUNE statement and autotune action set in SAS® Visual Data Mining and Machine Learning automatically tune hyperparameters of modeling algorithms by using a parallel local search optimization framework to ease the challenges and expense of hyperparameter optimization. This paper discusses the trade-offs that are associated with the different performance-enhancing measures and demonstrates tuning results and efficiency gains for each.

[Analyzing Text In-Stream and at the Edge](#)

Simran Bagga, SAS

As companies increasingly use automation for operational intelligence, they are deploying machines to read, and interpret in real time, unstructured data such as news, emails, network logs, and so on. Realtime streaming analytics maximizes data value and enables organizations to act more quickly. Companies are also applying streaming analytics to provide optimal customer service at the point of interaction, improve operational efficiencies, and analyze themes of chatter about their offerings. This paper explains how you can augment real-time text analytics (such as sentiment analysis, entity extraction, content categorization, and topic detection) with in-stream analytics to derive real-time answers for innovative applications such as quant solutions at capital markets, fake-news detection at online portals, and others.

[Harvesting Unstructured Data to Reduce Anti-Money Laundering \(AML\) Compliance Risk](#)

Austin Cook, SAS, Beth Herron, SAS

The financial services industry has called into question whether traditional methods of combating money laundering and terrorism financing are effective and sustainable. Heightened regulatory expectations, emphasis on 100% coverage, identification of emerging risks, and rising staffing costs are driving institutions to modernize their systems. One area gaining traction in the industry is to leverage the vast amounts of unstructured data to gain deeper insights. From suspicious activity reports (SARs) to case notes and wire messages, most financial institutions have yet to apply analytics to this data to uncover new patterns and trends that might not surface themselves in traditional structured data. This paper explores the potential use cases for text analytics in AML and provides examples of entity and fact extraction and document categorization of unstructured data using SAS® Visual Text Analytics.

[Invoiced: Using SAS® Text Analytics to Calculate Final Weighted Average Price](#)

Alexandre Carvalho, SAS

SAS® Contextual Analysis brings advantages to the analysis of the millions of Electronic Tax Invoices (Nota Fiscal Eletrônica) issued by industries and improves the validation of taxes applied. This paper highlights two items of interest in the public sector: tax collection efficiency and the calculation of the final weighted average consumer price. The features in SAS® Contextual Analysis enable the implementation of a tax taxonomy that analyzes the contents of invoices, automatically categorizes the product, and calculates a reference value of the prices charged in the market. The text analysis and the generated results contribute to tax collection efficiency and result in a more adequate reference value for use in the calculation of taxes on the circulation of goods and services.

[Using SAS® Text Analytics to Assess International Human Trafficking Patterns](#)

Tom Sabo, SAS, Adam Pilz, SAS

This paper showcases a strategy of applying SAS® Text Analytics to explore Trafficking in Persons (TIP) reports and apply new layers of structured information. Specifically, it is used to identify common themes across the reports, use topic analysis to identify a structural similarity across reports, identifying source and destination countries involved in trafficking, and use a rule-building approach to extract these relationships from freeform text. Subsequently, these trafficking relationships across multiple countries in SAS® Visual Analytics, using a geographic network diagram that covers the types of trafficking as well as whether the countries involved are invested in addressing the problem. This ultimately provides decision-makers with big-picture information about how to best combat human trafficking internationally.

Biomedical Image Analytics Using SAS® Viya®
 Fijoy Vadakkumpadan, SAS, Saratendu Sethi, SAS

Biomedical imaging has become the largest driver of health care data growth, generating millions of terabytes of data annually in the US alone. With the release of SAS® ViyaTM 3.3, SAS has, for the first time, extended its powerful analytics environment to the processing and interpretation of biomedical image data. This new extension, available in SAS® Visual Data Mining and Machine Learning, enables customers to load, visualize, process, and save health care image data and associated metadata at scale. This paper demonstrates the new capabilities with an example problem: diagnostic classification of malignant and benign lung nodules that is based on raw computed tomography (CT) images and radiologist annotation of nodule locations.

How to Build a Recommendation Engine Using SAS® Viya®
 Jared Dean, SAS

Factorization machines are a common technique for creating user item recommendations, there is evidence they generate double digit increases in engagement and sales. SAS has had recommendation methods for many years including market basket analysis, K-nearest neighbors (KNN), and link analysis, along with other techniques for creating a next best offer. This paper focuses on creating recommendations using factorization machines and SAS® Viya® 3.3. It describes each step of the process: 1) loading data into SAS Viya; 2) building a collaborative filtering recommendation model using factorization machines; 3) deploying the model for production use; and 4) integrating the model so that users can get on-demand results through a REST web service call. These steps are illustrated using the SAS Research and Development Library as an example. The library recommends titles to patrons using implicit feedback from their check-out history

We hope these selections give you a useful overview of the many tools and techniques that are available in the SAS AI platform.

Additionally, you can visit our [SAS AI Solutions](#) webpages to learn more about how these solutions are helping in some very cool crowdsourcing projects and how they can support your business needs.

We look forward to hearing from you – your questions as well as your experiences – so we together can continue to make AI pragmatic and results driven.

Saurabh Gupta, Director, Advanced Analytics and Artificial Intelligence Product Management



Saurabh Gupta, Director of Advanced Analytics and Artificial Intelligence Product Management, SAS Institute

During his tenure with SAS, Saurabh has overseen and driven product strategy for the Advanced Analytics, Artificial Intelligence, and Retail solutions portfolios. Saurabh graduated with a Ph.D. in Operations Management from The University of Texas at Austin. He has since devoted more than 19 years to specializing in large-scale systems analysis, design, and implementation in areas such as price optimization, supply chain management, and demand management. As a true advocate for leveraging his knowledge and skills to solve customer pain points, his work has received recognition from the journals of: *Management Science and Production* and *Operations Management*.

Paper SAS2246-2018
Navigating the Analytics Life Cycle with
SAS® Visual Data Mining and Machine Learning on SAS® Viya®

Brett Wujek, Susan Haller, and Jonathan Wexler, SAS Institute Inc.

ABSTRACT

Extracting knowledge from data to enable better business decisions is not a single step. It is an iterative life cycle that incorporates data ingestion and preparation, interactive exploration, application of algorithms and techniques for gaining insight and building predictive models, and deployment of models for assessing new observations. The latest release of SAS® Visual Data Mining and Machine Learning on SAS® Viya® accommodates each of these phases in a coordinated fashion with seamless transitions and common data usage. An intelligent process flow (pipeline) experience is provided to automatically chain together powerful machine learning methods for common tasks such as feature engineering, model training, ensembling, and model assessment and comparison. Ultimate flexibility is offered through incorporation of SAS® code into the pipeline, and collaboration with teammates is accomplished using reusable nodes and pipelines. This paper provides an in-depth look at all that this solution has to offer.

INTRODUCTION

With the ubiquity of data these days, companies are racing to ensure that they can apply analytics to derive the insight necessary to provide better products and services, and ultimately to keep pace with, or surpass, the competition. They know they need to “do machine learning,” but they frequently don’t really know what that entails. Their focus often turns directly to applying the powerful modeling algorithms to their data, resulting in individual eureka moments but neglecting the numerous phases of transforming data into business value in a sustainable manner.

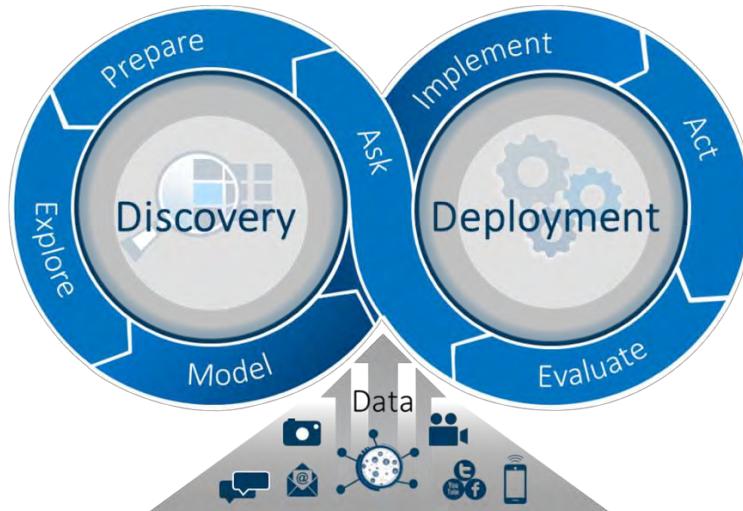


Figure 1. Phases of the Analytics Life Cycle

These important phases make up what is referred to as the analytics life cycle, as illustrated in Figure 1, which consists of the following:

- **Data ingestion:** consuming, merging, and appending data from potentially multiple data sources and formats
- **Data preparation:** cleaning, transforming, aggregating, and creating columns as necessary and appropriate to address the specified business problem
- **Exploration:** profiling, analyzing, and visualizing your data to gain initial insight and understanding of variable distributions and relationships

- **Modeling:** exercising feature engineering techniques, applying algorithms to identify segments and build representations for classifying new observations and making predictions, and assessing and tuning the generated models
- **Model deployment:** selecting champion models and promoting them for use in a production environment to aid in making effective business decisions
- **Model management:** maintaining a version-controlled repository of models, incorporating them into decision-making processes, monitoring their performance over time, and updating them as necessary to ensure that they are adequately and accurately addressing your business problem

Implementing and adhering to a process that accommodates the entire analytics life cycle is a significant undertaking, but a necessary one. Certainly, the public marketplace of analytics packages in open-source languages provides access to an ample supply of algorithms and utilities for data manipulation, exploration, and modeling. But typical business environments require more than individuals working on machine learning applications in silos and using a scattered collection of tools with little governance, lack of data and results lineage, inconsistent formats, collaboration bottlenecks, and hurdles to deployment. In the remainder of this paper, you will see how SAS Visual Data Mining and Machine Learning provides a comprehensive framework of capabilities to navigate this analytics life cycle through a seamless integration of interfaces that focus on each of the aforementioned phases, built on the foundation of SAS Viya. A case study that uses SAS Visual Data Mining and Machine Learning to address the problem of telecommunications customer attrition is presented in the Appendix.

THE FOUNDATION: SAS VIYA

An environment for end-to-end analytics relies on a solid foundation that can provide common access to data, analytics, and results in an efficient, consistent, and open manner. For SAS Visual Data Mining and Machine Learning, that foundation is provided by SAS Viya. SAS Viya is an extension of the SAS platform that offers a distributed, in-memory data access layer in which analytic “actions” can be performed in an efficient distributed and parallel manner through the SAS® Cloud Analytics Services (CAS) execution engine. Figure 2 illustrates the architecture, which is specifically designed to serve as an extensible and open framework in which data can be accessed from a variety of common sources and actions can be invoked in a language-agnostic fashion, and upon which custom and domain-specific applications can be established to exploit the in-memory efficiency and simple and common accessibility of data, actions, and results.

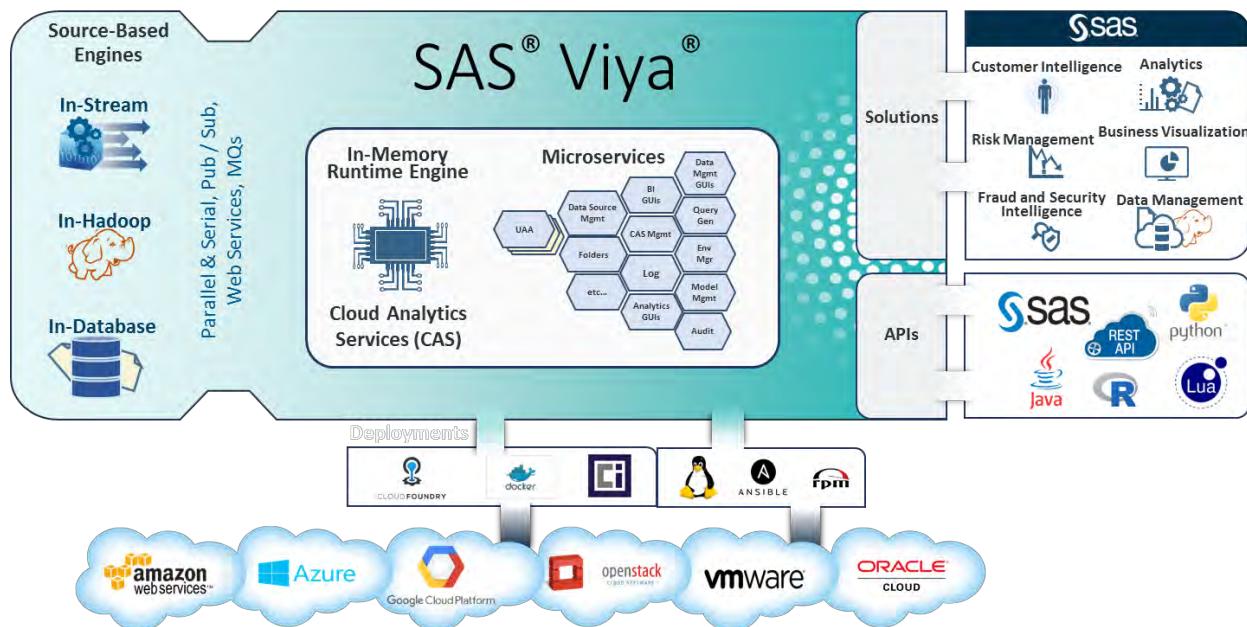


Figure 2. SAS Viya: An Extension of the SAS Platform

SAS Visual Data Mining and Machine Learning is one such application; it assembles a collection of data preparation and modeling actions that are presented through integrated interfaces that are specially designed for each phase of the analytics life cycle, as shown in Figure 3. When your work in one phase is complete, you can directly progress to the next phase, avoiding any hassle (and error-prone process) of transferring (and possibly translating) your data or results, or of launching new applications independently. Because the analytics are performed by invoking actions in CAS, the data preparation and modeling functions can also be executed by writing programs in SAS or other languages for which an API wrapper has been written (Python, R, Java, Lua, and REST). A good example of how you can work on a particular machine learning application across multiple interfaces and programming languages is offered in Wexler, Haller, and Myneni (2017). This is all made possible by SAS Viya providing the common data access layer and open access to a consistent set of analytics actions.

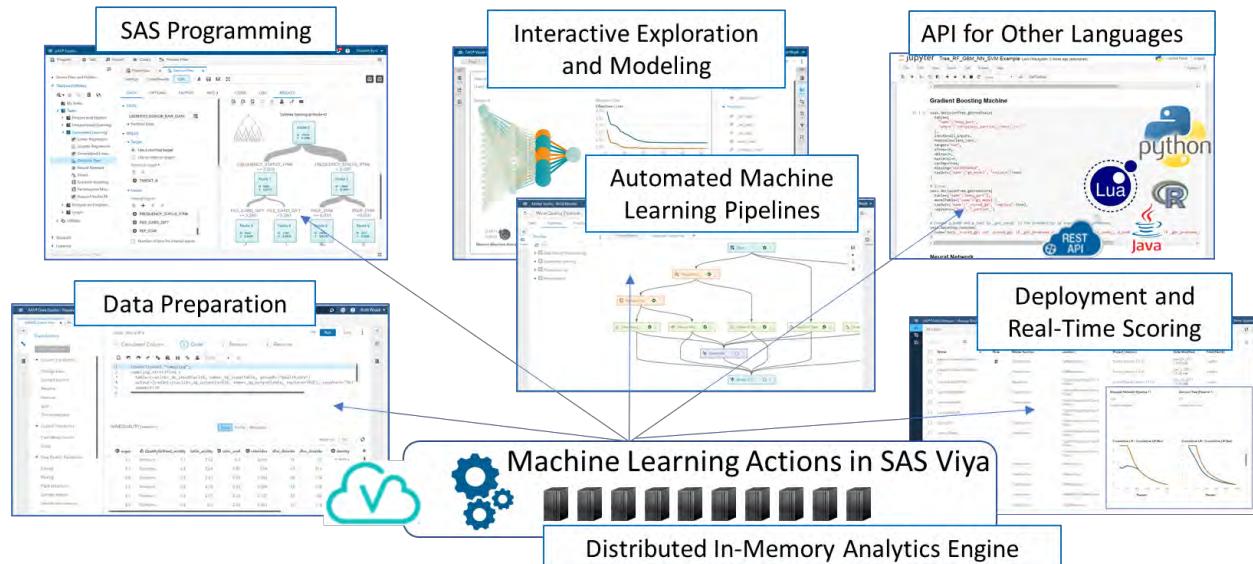


Figure 3. SAS Visual Data Mining and Machine Learning Capabilities and Interfaces

One means of employing the capabilities that comprise SAS Visual Data Mining and Machine Learning is through an integrated collection of actions in a unified web interface that is designed specifically to facilitate the end-to-end analytics life cycle, as depicted in Figure 4. The remainder of this paper navigates through the analytics life cycle with SAS Visual Data Mining and Machine Learning via the user interfaces that are associated with these actions.

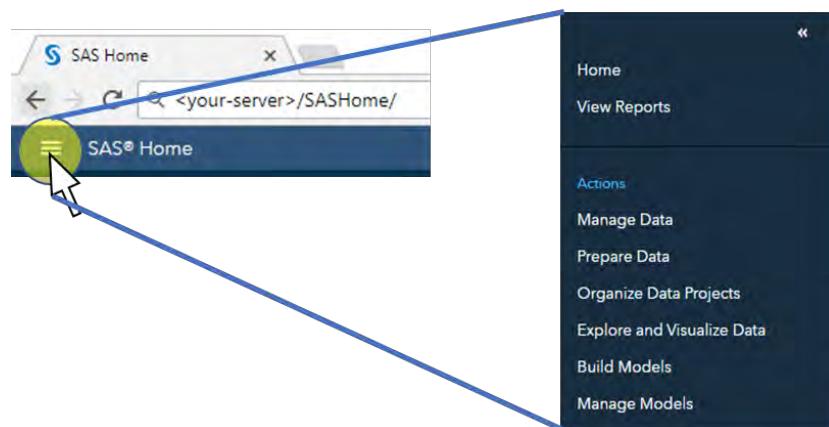


Figure 4. Menu of Actions to Access SAS Visual Data Mining and Machine Learning Capabilities

DATA INGESTION AND PREPARATION

Machine learning applications should be developed and evolve as solutions to well-defined business problems. That is, assuming you have (or can get) the necessary data, what are the most important questions you would like answered to add value to your organization? This is the “Ask” phase of the analytics life cycle shown in Figure 1, and it goes hand-in-hand with collecting the requisite data, identifying the necessary analytical operations, and ensuring that your data are in an appropriate form for these analytics. Although a software platform cannot resolve the “Ask” for you, it *can* support it by the accommodations it provides for ingesting and preparing data for the desired analytical operations in the “Prepare” phase of the analytics life cycle, as shown in Figure 5.

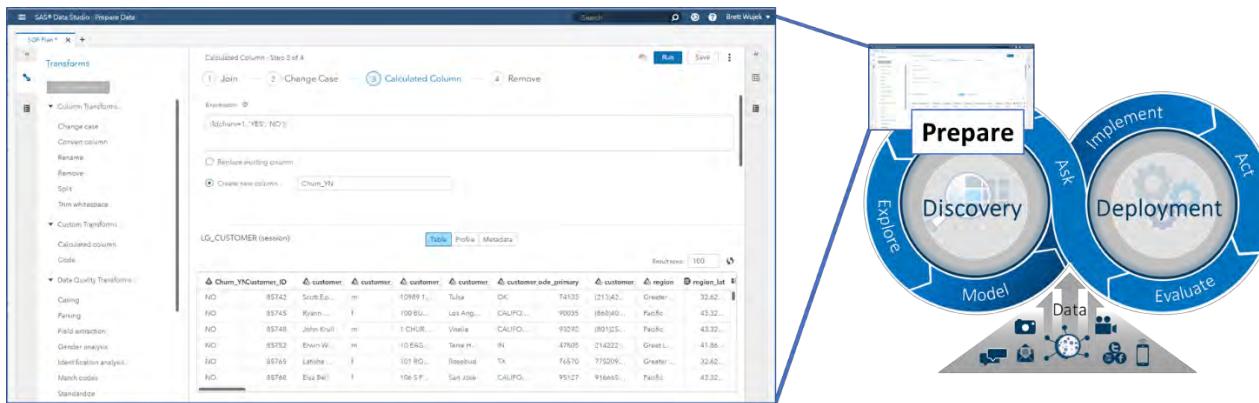


Figure 5. Data Preparation Using SAS® Data Studio in SAS Visual Data Mining and Machine Learning

CONSUMING DATA FROM VARIOUS SOURCES

SAS Visual Data Mining and Machine Learning provides built-in conveniences for browsing available data and importing data from various sources as necessary. A common data browser is used in all interfaces wherever a data table needs to be selected (see Figure 6). For extended data management capabilities, an enhanced form of the data browser can be added to your environment, offered as a dedicated Data Explorer interface, which is accessible from the **Manage Data** action in the actions menu.

Figure 6. Browsing and Loading Data

To prepare, explore, and perform analytics on data in SAS Viya, the data must be loaded into memory as a CAS table. The data browser displays data tables that are available to use immediately (data sets that have been loaded into CAS tables), denoted by the  icon next to the table name. Data can be made available by defining connections to new **Data Sources** by clicking the “Connect” button (), and referencing data sets that reside in those data sources. SAS Viya supports several types of data sources by using data connectors (depending on SAS/ACCESS® licensing), including the following:

- **File system:** DNFS, HDFS, Path
- **Database:** DB2, Hadoop Hive, Impala, LASR, ODBC, Oracle, PostgreSQL, Teradata

Once a data source is defined, a CAS library (caslib) serves as a reference to it and is presented for you to browse available tables. Although you can browse *all* tables (including SAS data sets) that reside in the data source, you can select only tables that are loaded as in-memory CAS tables to use within the application.

 **Tip:** *To load a data set that resides in a specified data source but is not yet loaded (that is, it has an icon other than  next to it), right-click it and select Load.*

You can also import data from local files, such as the commonly used CSV (comma-separated values) format or other text files that contain data in a tabular format, or directly from social media feeds such as Twitter, Facebook, Google, and YouTube. The main thing to keep in mind is that a data set must be loaded into memory as a CAS table before you can work with it.

For a selected table, the data browser displays all the column names along with their corresponding data types, in addition to information about the size of the table, as shown in Figure 6. You can run a profile of the table to get an initial indication of the cardinality, number of missing values, and basic descriptive statistics for each variable. The profile provides some insight as to what type of data preparation you might need to exercise before applying analytical operations on or modeling the data.

TRANSFORMING AND ENHANCING YOUR DATA

Data preparation is such an important and necessary step in machine learning applications (Wujek, Hall, and Gunes 2016) that you will find capabilities to transform and augment your data in various forms throughout different interfaces in SAS Visual Data Mining and Machine Learning. Often, interactive visual inspection of distributions and other aspects of the data is necessary in order to understand which analytical transformations are required, and other specialized forms of data manipulation, such as feature engineering techniques, are more closely associated with the model building phase. For data preparation to be done in a systematic and repeatable fashion so that it can be applied consistently to new data in the future, SAS Visual Data Mining and Machine Learning provides a powerful and convenient interface, SAS Data Studio, for preparing your data. SAS Data Studio enables you to build a data plan that consists of a sequence of well-defined, repeatable steps that apply transforms to the source data table that is loaded. These transforms are organized in the following categories:

- **Column Transforms** to modify the values in existing columns in common ways
- **Row Transforms** to filter rows on the basis of variable values or to create columns through transposition
- **Multi-input Transforms** to join/merge or append tables
- **Data Quality Transforms** to standardize values and apply common data cleansing operations by using a SAS® Quality Knowledge Base
- **Custom Transforms** to calculate new columns by using simple expressions or custom code

The out-of-the-box transforms provide a convenient way to quickly transform (and clean) the values in columns and create new columns through aggregation and calculations. For any data preparation actions that are not directly available as transforms, the **Code** transform (one of the **Custom Transforms**) provides ultimate flexibility by enabling you to write SAS DATA step or CASL code to prepare your data as necessary.

Tip: When writing code for the **Code** transform in order to prepare data, you must use the variables `_dp_inputCaslib`, `_dp_inputTable`, `_dp_outputCaslib`, and `_dp_outputTable` to refer to the input and output tables.

handset	Customer_ID	churn	avg_arpu_3m	acct_age	contracts_ltd	credit_cla	sales_cha	region	state	zip
APPLE	250...	0	\$51.78	46	5	prime	Retail	Pacific	CA	94012
HTC	251...	0	\$60.00	25	1	near pri...	Brande...	Midwest	MO	64101
APPLE	251...	0	\$63.99	58	5	near pri...	Retail	Great Lu...	MI	48101
APPLE	251...	0	\$51.78	46	5	prime	Retail	Southwest	AZ	85012
MOTOR...	251...	0	\$46.66	42	6	near pri...	Indirect	South	TN	37101
APPLE	251...	0	\$55.30	44	5	prime	Nationa...	Pacific	WA	98101

Figure 7. Building and Applying a Data Preparation Plan in SAS Data Studio

Each transform that is added as a step in the data plan must be defined and then run so that an updated version of the table is available for a subsequent step. The plan maintains a reference to the unaltered source data table while it creates and updates a new table as a result of applying the steps. Information about the source table can be seen on the left, and information about the result table can be viewed on the right. Profiles of the source and result tables can be run to view information about the variables.

Tip: If the result table is not as expected or desired, you can roll back the list of steps from last to first by clicking the undo button .

USING YOUR PREPARED DATA

A major advantage of SAS Visual Data Mining and Machine Learning is the ability to seamlessly navigate from one phase of the analytics life cycle to another. Once you have defined the data preparation plan, you can save the plan so that it can be applied to new data tables, and you can use the actions menu () to progress directly to other phases that will use the result table, as shown in Figure 8. To continue navigating through the analytics life cycle, you can select **Explore and Visualize Data** to get a good sense of the nature of your data and the relationships among the variables.

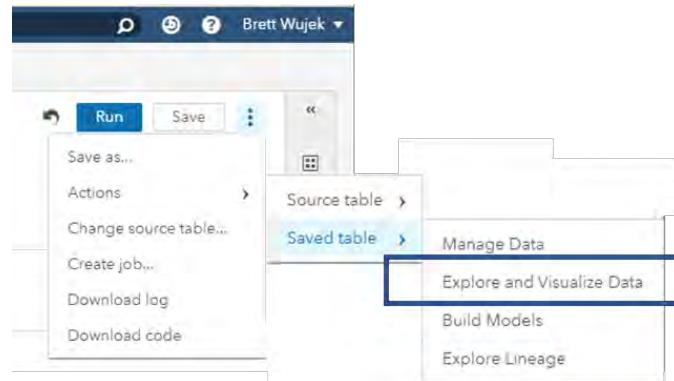


Figure 8. Actions in SAS Data Studio for Navigating through the Analytics Life Cycle

INTERACTIVE EXPLORATION AND MODELING

Extracting value from your data requires a certain degree of understanding the data, and although descriptive statistics and other forms of profiling are useful tools for this, there is no substitute for exploring your data interactively in a visual manner. The “Explore” phase of the analytics life cycle sets the stage for more in-depth analysis and modeling, enabling you to gain some initial insights from variable distributions and relationships, and providing a realization of the potential payoff that can be expected of predictive modeling.

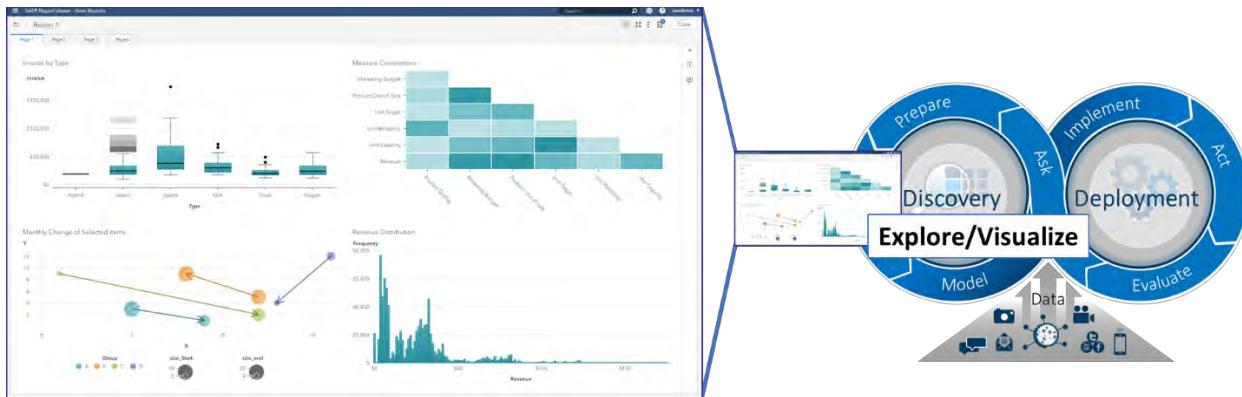


Figure 9. Exploring Your Data in SAS Visual Data Mining and Machine Learning

EXPLORING YOUR DATA

SAS Visual Data Mining and Machine Learning provides very powerful and intuitive visual exploration capabilities in the SAS® Visual Analytics interface. You can very quickly view the nature of your variables by using bar charts of distributions, and you can understand the relationships among variables by using objects such as scatter plots and correlation matrices. You can also get a sense for observational groupings by using crosstabulation tables (crosstabs), parallel coordinate plots, and clustering algorithms.

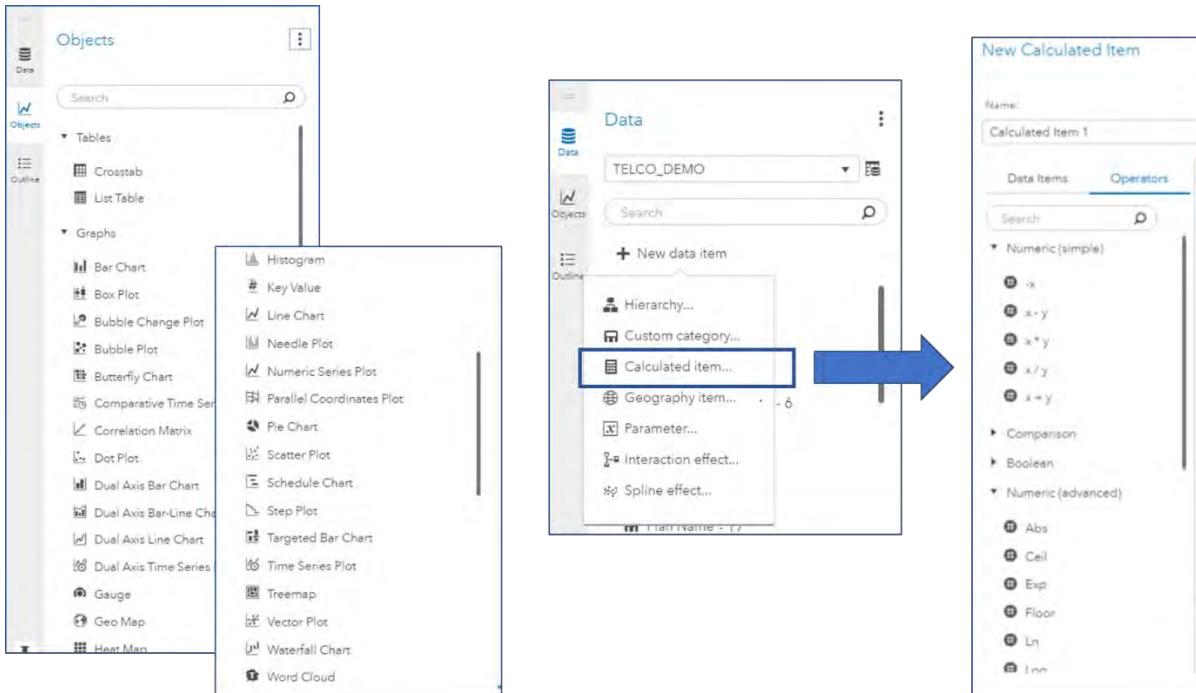


Figure 10. Visualization and Data Preparation in SAS Visual Analytics

Based on insight gained by exploring your data, or possibly from prior domain knowledge, you might need to transform columns or augment your data with new variables that are functions of existing variables. As previously stated, data preparation occurs in many forms in different phases of the analytics life cycle. During interactive exploration, you can create new data items very simply (without any programming) by selecting from a wide array of mathematical, comparison, date and time, text, and aggregation operators to build expressions that generate new columns. Persistent attention to the representation of your data leads to better, more meaningful predictive modeling results.

BUILDING PREDICTIVE MODELS

Ultimately, moving from *descriptive* analytics (understanding the nature of your data in terms of historical behavior and trends) to *predictive* analytics (realizing what might happen in the future based on the historical data) involves building models to represent the relationships between input variables and a target of interest. Using such models to classify new observations or predict target values is, of course, the focus of machine learning. One of the goals of SAS Visual Data Mining and Machine Learning is to offer these modeling capabilities in different forms that can be consumed by users who have various levels of expertise. To that end, you can build several different types of predictive models in SAS Visual Analytics in an interactive fashion with no programming required.

A crucial step in building predictive models is to ensure that you hold out data from the training process to honestly assess the accuracy of the model on data that was not seen during training. For this purpose, SAS Visual Analytics enables you to define partitions in your data by one of the following methods:

- Select a category variable that has two or three levels and select **Set as partition column**.
♀ **Tip:** Convert a Measure variable to Category in order to select it as the partition variable.
- Click the  button next to the data source and select **Add partition data item** to create a new column to use to define the partitioning.

Models can be trained without a validation partition, but doing so is highly discouraged because overfitting to the training data will most likely occur and your model will not generalize well (that is, accuracy of predictions will diminish).

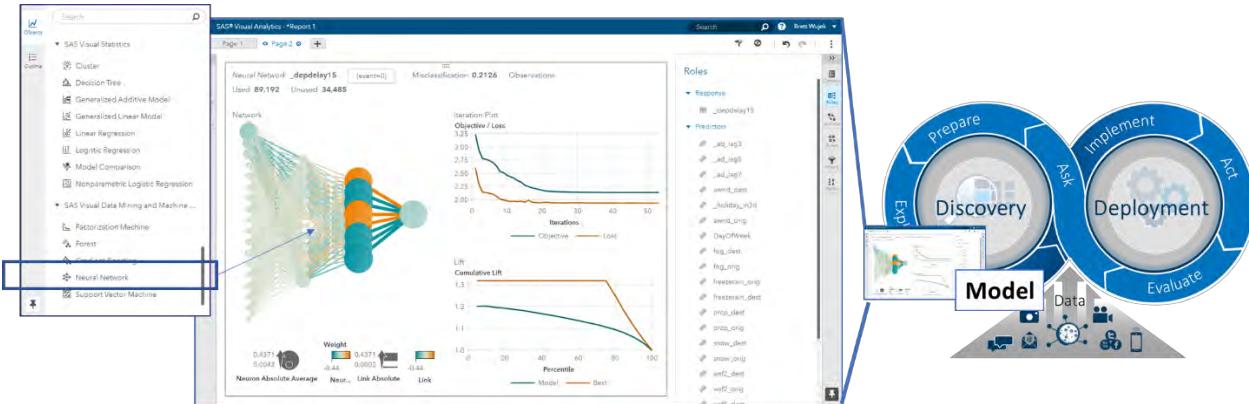


Figure 11. Interactive Modeling Using SAS Visual Analytics in SAS Visual Data Mining and Machine Learning

As shown in Figure 11, various objects for building predictive models are available in SAS Visual Analytics; the more modern machine learning algorithms fall under SAS Visual Data Mining and Machine Learning, whereas the basic regression techniques and decision tree fall under SAS® Visual Statistics. Training a model in SAS Visual Analytics is as simple as dragging a modeling object onto a page (or double-clicking it) and selecting the **Response** (target variable to predict) and **Predictors** (input variables) in the **Roles** tab on the right, as shown in Figure 11. You also need to specify the variable that serves as the **Partition ID** if one has been defined. Once you have selected the desired roles for your model, you can conveniently create other types of models that have the same roles by right-clicking and selecting **Duplicate as**.

♀ Tip: To create the model on a new page, press the Alt key when you right-click.

The **Options** tab presents many algorithm settings, called *hyperparameters*, that can be adjusted to drive the training process. Some models provide right-click menu options for editing the model hyperparameters; for example, you can add, remove, or edit hidden layers by right-clicking a neural network diagram. Modifying the hyperparameters and immediately observing the resulting change in model accuracy through various metrics and assessment plots gives you an interactive means of gaining insight regarding the types of models you can build. Although you can manually explore different combinations of these settings, the number of all possible configurations makes manual exploration infeasible. To address this issue, for several algorithms you can select **Autotune** to instruct the software to automatically adjust the hyperparameters by using an intelligent search strategy to find the best model, as described in Koch et al. (2017). Since autotuning requires training numerous candidate models, invoking it disables interaction with this modeling object until the process completes, which could take several minutes. However, several measures are taken in the autotuning implementation to make the process as efficient as possible by managing parallel use of computing resources and using early stopping techniques (Koch, Wujek, and Golovidov 2018).

COMPARING AND USING YOUR MODELS

Training effective predictive models is somewhat of an art. Beyond applying different data preparation steps, employing feature engineering techniques, and finding the best hyperparameter settings to use, assessment of a model can be carried out using several different metrics. As you interactively build different models, often with different modeling algorithms, direct comparison can be cumbersome as you look at each model independently. SAS Visual Analytics provides a **Model Comparison** object that enables you to easily compare your models side-by-side in a variety of ways by presenting multiple standard assessment plots, different metrics, and cutoff and percentile selections. Models that have the same variables defined for the roles can be selected to include in the comparison.

♀ Tip: Modifying and retraining a model after the **Model Comparison** object has been created requires that you create a new **Model Comparison** object; it cannot be updated.

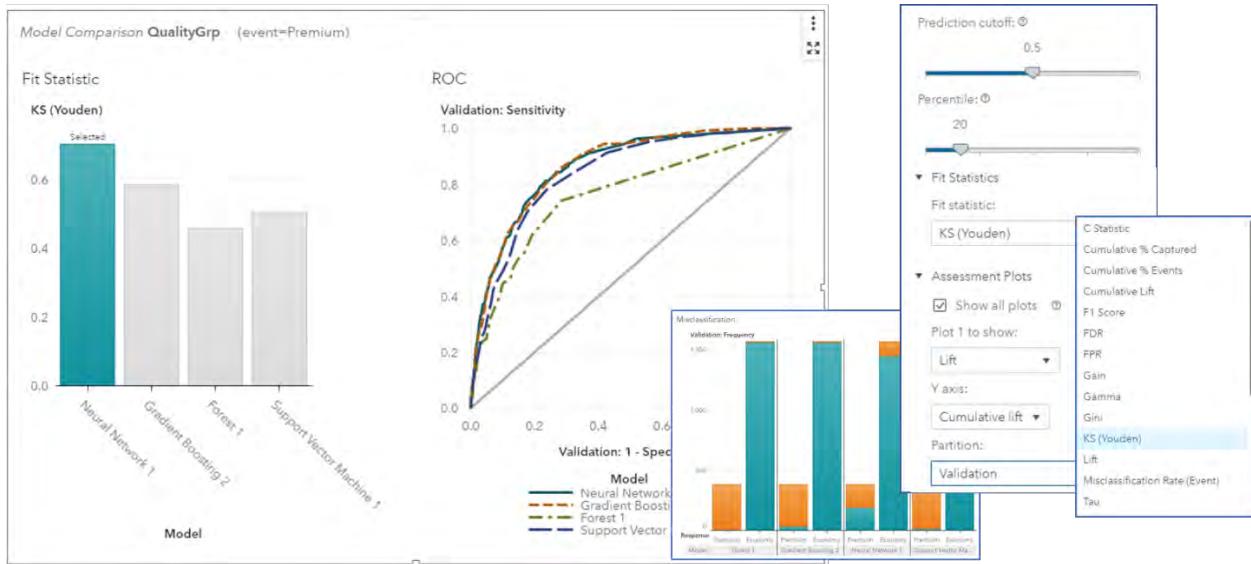


Figure 12. Comparing Models in SAS Visual Data Mining and Machine Learning

Once you have identified a model that you want to use for scoring new observations, you can right-click on the model and select **Export model**. This generates the score code for the model and downloads it for you to use as desired. Models that use the binary “ astore” (analytic store) format save the model to the Models caslib in addition to downloading the SAS code that can be used to invoke the astore model for scoring.

Interactively building and assessing models within SAS Visual Analytics gives you a sense of the level of predictive power you can hope to achieve and the effect that different algorithms and their associated hyperparameters have in building accurate predictive models. In some cases, the resulting models are sufficient for use in your production environment. However, quite often they are best used as starting points for constructing more detailed and sophisticated machine learning applications by enhancing them to incorporate advanced feature engineering techniques and effective ensembling methods. Building pipelines to represent the flow of data through sequences of connected nodes that apply these techniques is an effective way of automating the process and ensuring repeatability. In SAS Visual Analytics, you can right-click on any model and select **Create pipeline** to progress to the next level of modeling, as depicted in Figure 13.

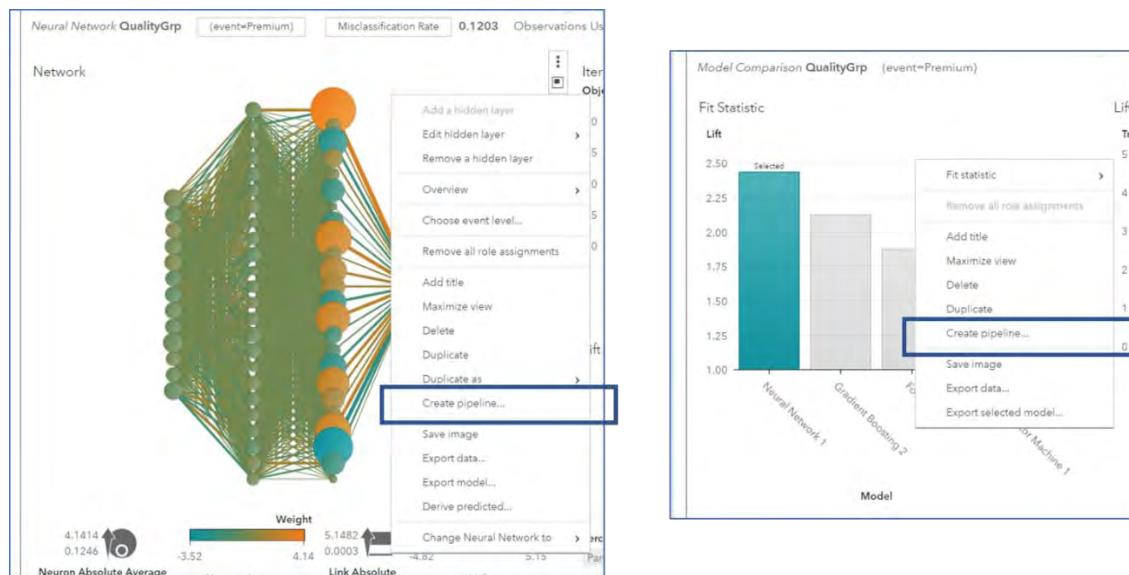


Figure 13. Taking Interactive Modeling to the Next Level by Creating Pipelines

AUTOMATED MODELING PIPELINES

Pipelines serve as self-documenting, automated, repeatable processes; they offer flexibility in the steps taken to build, compare, and ultimately choose the model for your business problem. Overall, they are tremendous productivity enhancers. To take you beyond interactive, ad-hoc modeling, SAS Visual Data Mining and Machine Learning offers a pipeline-centric, collaborative modeling environment in Model Studio. This feature-rich interface enables you to build automated processes that exercise feature engineering techniques, to apply algorithms to identify segments and build representations for classifying new observations and making predictions, and to assess and compare the generated models.

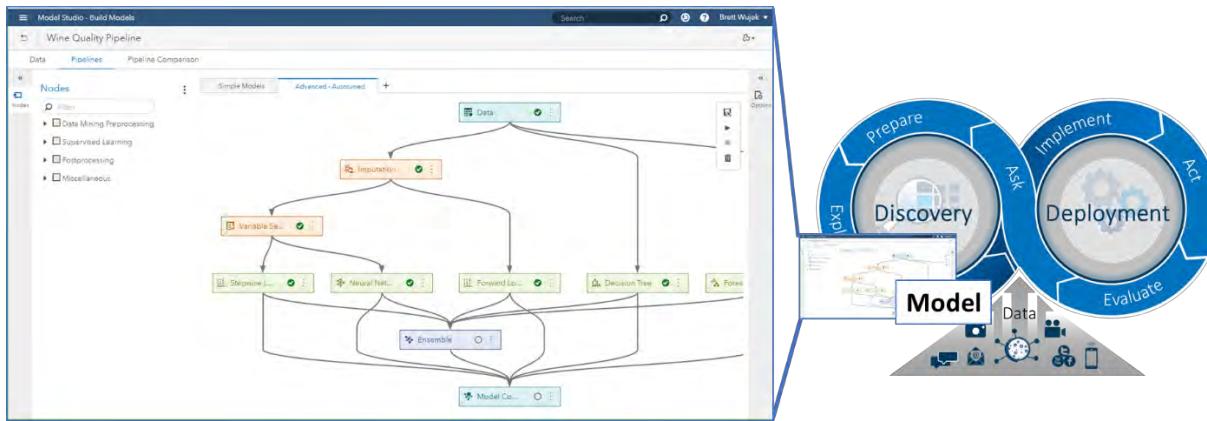


Figure 14. Automated Modeling Using Model Studio in SAS Visual Data Mining and Machine Learning

PROJECTS

The primary object for defining and managing pipelines for machine learning applications in Model Studio is a *project*. The Projects page serves as the top-level home page in Model Studio; it presents all the projects that you have created or that are accessible to you, either as a table or as a display of tiles. Since Model Studio is an interface that enables you to build pipelines for other domains—namely forecasting (for SAS® Visual Forecasting) and text analytics (for SAS® Visual Text Analytics)—you might see a mixture of projects of different types listed, depending on the products you have licensed. The color of the tile (or the Type column in the table view) indicates the type of project; for “Data Mining and Machine Learning” projects, the tile also presents a thumbnail image that corresponds to the type of model that is determined to be the *champion* (best) model for that project. The champion model is determined by assessing and comparing all models that have been trained in the project based on a specified metric.

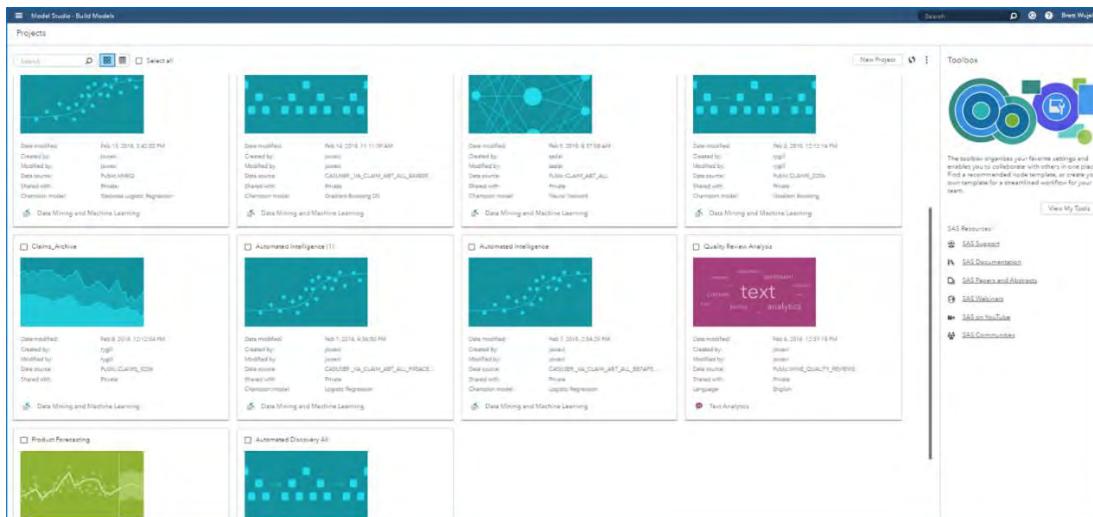


Figure 15. Model Studio Projects Page

By default, projects are private, meaning they are visible only to the creator and administrators, and editable only by the creator. However, Model Studio is designed and developed to be a collaborative environment, so projects can be shared with defined groups by selecting the project, clicking **Share** in the actions menu, and specifying the group with which it is to be shared, as illustrated in Figure 16. You can share a project as read-only or you can let others edit it; Model Studio ensures that only one person can edit a project at a time.

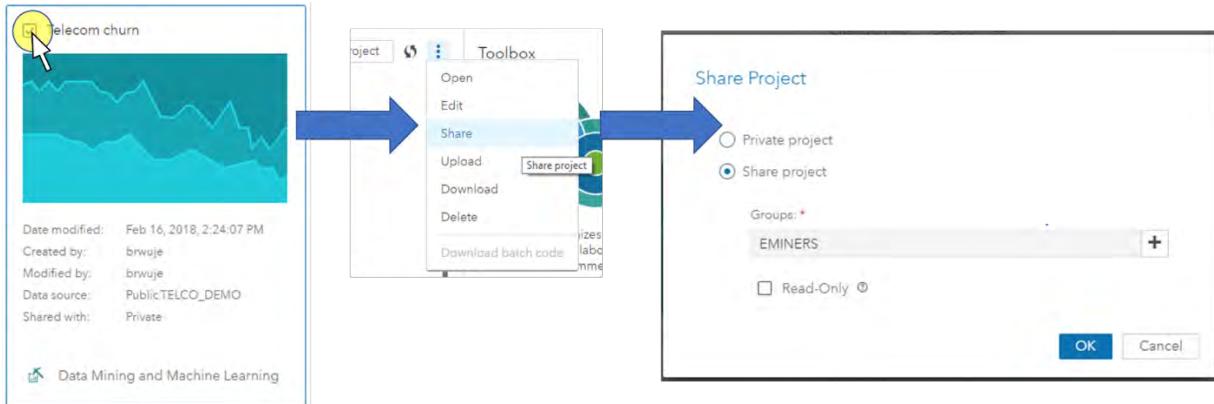


Figure 16. Sharing Model Studio Projects in SAS Visual Data Mining and Machine Learning

To archive projects offline or transfer them among servers, you can download a selected project by using the actions menu (⋮). The project is downloaded as a ZIP file that contains JSON files with all the information necessary to recreate the project. When uploading a project ZIP file, you simply need to specify the data source by selecting a CAS table available on the CAS server.

A well-defined and completed project can serve as a mechanism for generating a production-worthy model that is deployed to support making business decisions. Given that models can become “stale” (decay in accuracy) over time, you might need to re-execute your project, using new source data to generate a new model to consider for production. To avoid the need to use the Model Studio user interface for this, you can download batch code that contains the necessary RESTful API calls—invoked from SAS, Python, or REST (representational state transfer) code—to invoke the services to re-execute the project and generate new models. The code also provides API calls to check the status of the project execution and to obtain the new champion model after it is complete.

DATA DEFINITION

The first step in building models is to understand and define the metadata for variables that you will be using within the pipelines in your project. On the **Data** tab of Model Studio, you will find high-level summary statistics for each variable within your table, such as the count of unique levels, percentage of missing values, and the minimum, maximum and mean for your continuous variables. These metrics determine the default measurement level and roles that are assigned to each variable. For example, variables that contain more than 50% missing values are automatically assigned a role of **Rejected** and excluded from your analysis. All numeric variables that have more than 20 levels are assigned a measurement level of **Interval**.

Although Model Studio automates the generation of this metadata, you can also interact with this metadata and make modifications to the default definitions that are provided. To do so, highlight one or more variables within your table and select **Edit variable**.

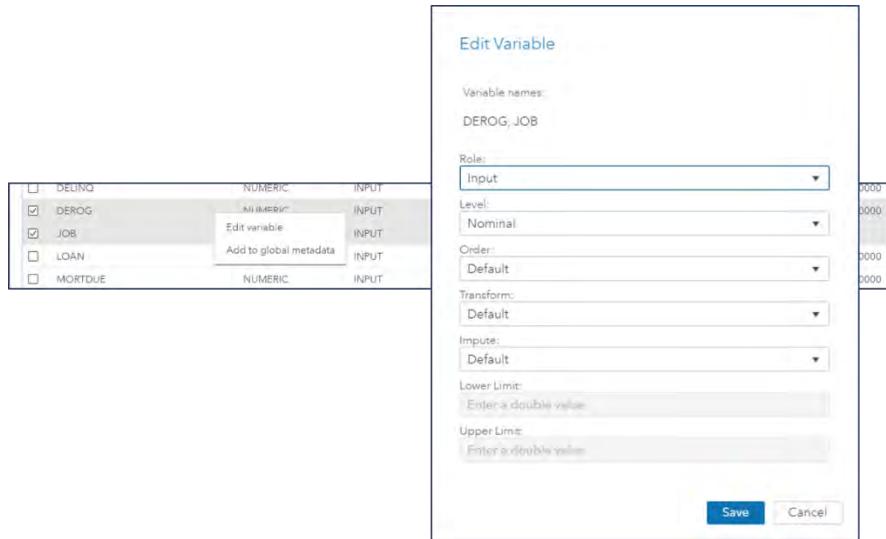


Figure 17. Editing Variables on the Data Tab in Model Studio

♀ **Tip:** One target variable and at least one input variable are required to run a pipeline in Model Studio. Once a target is defined and a pipeline has been executed, the target cannot be modified.

You can modify the default settings for your metadata, and you can also define variable-specific imputation and transformation strategies that can be used within your pipelines. The imputation and transformation methods that are offered depend on the measurement level of the variable you are editing. It is important to note that the imputation and transformation methods defined here are not applied to your variables until you include the corresponding Imputation or Transformation nodes in your pipeline, as described in the next section. When these nodes are executed, the variable-specific methods are applied where defined and node-specific methods are applied to all other variables.

♀ **Tip:** To see additional columns in the **Data** tab in Model Studio, you can customize the display by using the **Manage Columns** button (⋮) found in the upper right corner of the table itself. This button enables you to add or remove columns from the table and to designate their order.

There are many cases in which data definitions for a variable might span multiple projects or even multiple data sources themselves. You can avoid defining this information in each instance by selecting **Add to global metadata** from the actions menu (⋮) in the upper right. Each time a new project is created and metadata are generated, Model Studio checks for variables that match global metadata definitions by name (case-sensitive) and type, and it pulls in and reuses the information that is stored in the global metadata when it assigns values. Global metadata that has been defined can be managed in the Toolbox, which is accessible from the Projects page in Model Studio.

PIPELINES

As previously mentioned, Model Studio enables you to build automated process flows, called pipelines, that accommodate all the steps that are involved in a typical machine learning application. The **Pipelines** tab of Model Studio presents a visual representation of your pipelines as you construct them using “nodes.” All pipelines start with a Data node to inject the project data into the flow and perform any specified partitioning (done only once for the project), and each step in a pipeline is represented by a node from one of the following categories:

- **Data Mining Preprocessing:** Nodes for manipulating and studying the data before building any models

Note: The “Data Mining” prefix is used to differentiate these nodes from preprocessing nodes for other domains such as forecasting and text analytics. The prefix is omitted for the remainder of this paper.

- **Supervised Learning:** Nodes for building models to predict your specified target
- **Postprocessing:** Nodes for performing operations on models that are built by upstream nodes; currently this is dedicated to building ensemble models
- **Miscellaneous:** Nodes for various useful auxiliary capabilities

The specific available nodes within these categories are listed in Figure 18.

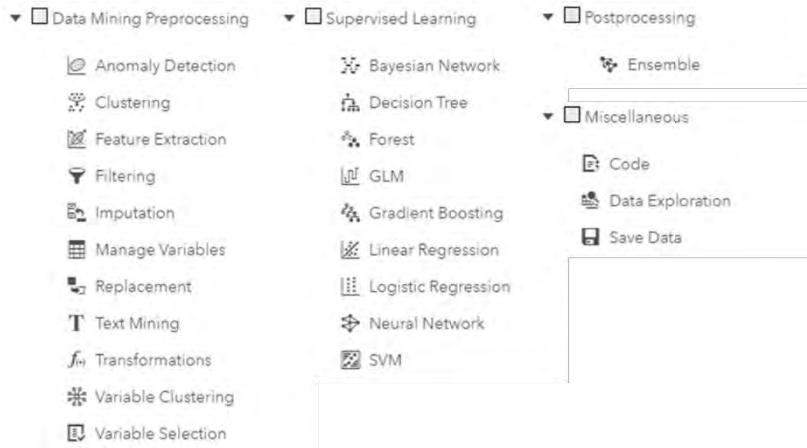


Figure 18. Categories of Nodes Available to Build Pipelines in Model Studio

Pipeline Construction

Instead of just providing an empty canvas that allows free-form pipeline construction in which nodes are added and connected in any desired fashion, Model Studio takes the deliberate approach of enforcing that nodes be connected only in a meaningful fashion, based on their categories. Supervised Learning nodes cannot be added before Preprocessing nodes, Postprocessing nodes must come after Supervised Learning nodes, Supervised Learning nodes cannot be run in sequence (that is, in the same branch), and the Data Exploration and Save Data nodes are terminal nodes. These rules ensure that the logic of your pipelines remains intact. Pipelines can take full advantage of the distributed execution environment of SAS Viya by executing different independent nodes in parallel branches so that they can execute simultaneously.

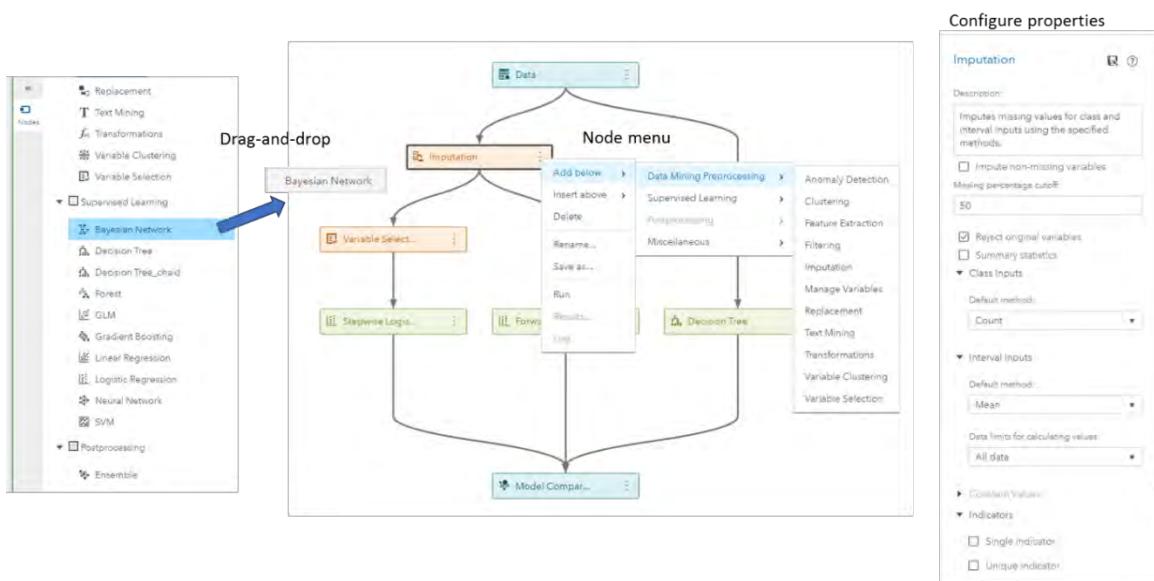


Figure 19. Constructing Machine Learning Pipelines in Model Studio

Creating pipelines is as simple as dragging nodes onto the pipeline canvas on top of the node that you want it to follow, or using the node menu (⋮) to insert nodes below or above an existing node. The pipeline rules described earlier ensure that you are adding nodes in appropriate locations, and parallel branches are automatically created when multiple nodes are added after a node. Because the primary goal of Model Studio is to automate and facilitate building multiple candidate models and comparing them to identify deployment-worthy models, all Supervised Learning nodes are automatically connected to a Model Comparison node that assesses the models and presents the results for comparison. This is discussed further in the section “Model Comparison.”

While building pipelines from scratch offers the flexibility to introduce whatever logic is deemed necessary for your machine learning application, existing pipelines can often serve as good starting points to avoid continually building similarly structured flows. Pipelines that have proven to be effective at building good models for one problem, or data set, can serve as “templates” to be applied to another problem. Model Studio comes supplied with pipeline templates for a number of scenarios and levels of modeling. When you add a new pipeline to your project, you can select a template as a starting point and then modify it and configure the node properties as desired.

Template Name	Description	Owner	Last Modified
Advanced template for class target	Extends the intermediate template for class target with neural network, forest, and gradient boosting models, as well as an ensemble.	SAS Pipeline	Jan 9, 2018, 4:38:30 PM
Advanced template for interval target with autotuning	Advanced template for interval target with autotuned tree, forest, neural network and gradient boosting models.	SAS Pipeline	Jan 9, 2018, 4:38:41 PM
Advanced template for interval target	Extends the intermediate template for interval target with neural network, forest, and gradient boosting models, as well as an ensemble.	SAS Pipeline	Jan 9, 2018, 4:38:49 PM
Basic template for class target	A simple linear flow: Data, Transformations, Imputation, Logistic Regression, Model Comparison.	SAS Pipeline	Jan 9, 2018, 4:38:53 PM
Basic template for interval target	A simple linear flow: Data, Transformations, Imputation, Linear Regression, Model Comparison.	SAS Pipeline	Jan 9, 2018, 4:38:56 PM
Blank Template	A Data Mining pipeline that contains only a data node.	SAS Pipeline	Jan 9, 2018, 4:38:57 PM
Intermediate template for class target	Extends the basic template with a stepwise logistic regression model and a decision tree.	SAS Pipeline	Jan 9, 2018, 4:39:02 PM
Intermediate template for interval target	Extends the basic template with a stepwise linear regression model and a decision tree.	SAS Pipeline	Jan 9, 2018, 4:39:11 PM

Figure 20. Pipeline Templates in Model Studio

Collaboration

SAS Visual Data Mining and Machine Learning was designed and developed with the mindset that data mining and machine learning projects are a collaborative effort, and that these projects often produce artifacts that are useful to other projects. The pipeline templates that are included in Model Studio are a basis that you can extend by saving pipelines that you create, allowing them to be used to create new pipelines in other projects, possibly by other users. To save a pipeline so that it can be used in other projects, click the Save button (💾) in the pipeline toolbar and provide a name and description for the template. The entire pipeline structure and all properties of all the nodes are retained so that new instances will start as exact copies of this template. This template will appear along with the predefined templates (with the corresponding creator specified) when you select a template to create a new pipeline. You can manage templates in the Toolbox, which is accessible from the Projects page in Model Studio.

Beyond sharing and reusing pipelines as templates, often you will find that a certain configuration of a node is very effective and you would like to use it in other pipelines or share it with others. For example, if you typically like a certain method and settings for feature extraction, you can configure a Feature

Extraction node and save a copy of it to the Toolbox by clicking the Save button (■) at the top of the Properties panel for that node. A copy of that node will then reside in the Toolbox and will be accessible in the appropriate category in the Nodes panel for selection and use in building pipelines.

Some Noteworthy Nodes

A detailed discussion of all available nodes is beyond the scope of this paper. Most of the nodes serve as convenient interfaces to underlying CAS actions that are associated with common data mining and machine learning capabilities. However, the following handful of nodes are highlighted because they provide specialized functionality to supplement and enrich your pipelines.

Manage Variables

As described previously, the **Data** tab enables you to specify information (metadata) about how your variables should be used in the project. However, often you want to specify different metadata in order to use the variables in special ways in different branches of a pipeline, or for different pipelines. For example, you might want to use different inputs for different models, or you might want to apply different transformations from those that are defined on the **Data** tab. The Manage Variables node enables you to do just that. After you insert a Manage Variables node, you must initially execute it in order to allow it to import the current metadata information; you can then view an editor that enables you to modify the variable metadata.

♀ **Tip:** Once a pipeline has been run in a project, metadata for the variables of the project cannot be modified on the **Data** tab. To modify the variable metadata for a specific pipeline, use the **Manage Variables** node within that pipeline.

Data Exploration

In-depth interactive exploration is best done in SAS Visual Analytics as discussed in the section “Exploring Your Data.” But often you are working on a pipeline in Model Studio and you want to get a sense of some intermediate form of your data. The Data Exploration node is a Miscellaneous node that displays summary statistics and plots for variables in the data table that is provided by the preceding node. The Data Exploration node selects a subset of variables to provide a representative snapshot of the data. Variables can be selected to show the most important inputs, or to indicate “suspicious variables” (that is, variables that have anomalous statistics). You can use the Data Exploration node to identify good candidate variables for inclusion in predictive models as well as variables you might want to exclude. This node can suggest variables that might require transformation (for example, variables that have skewed distributions) or imputation of missing values.

♀ **Tip:** To explore a model's predicted values, connect the **Data Exploration** node to the **Supervised Learning** node that produces that model.

Save Data

By default, the table that is produced by a node in a pipeline is temporary; it exists only for the duration of the run of the node and has local session scope. You can connect a Save Data node to another node in order to save the output table to disk in the location that is associated with the specified output library. This table can then be used later by other applications for further analysis or reporting (for example, in SAS Visual Analytics).

♀ **Tip:** To allow the saved table to be seen in other CAS sessions, select the option to **Promote table**.

Ensemble

Quite often, the most accurate predictions are not provided by a model that is trained from a single instance of an algorithm, but instead are provided by combining the predictions of multiple models into an

ensemble model. The Ensemble node is a Postprocessing node that enables you to create a new model by using a functional combination (aggregation) of posterior probabilities (for class targets) or predicted values (for interval targets) from multiple models in a pipeline. You can add an Ensemble node after any Supervised Learning node and then continue to add other existing models to it by using the **Add Models** option in the node menu (⋮). General model assessment statistics are provided in the results, and the generated ensemble model is treated just like other models in terms of comparison and potential selection as a champion. Score code for the Ensemble node is produced by combining the score code (DATA step or analytic store) of the constituent models.

Code

The SAS language is very expansive and contains many useful and powerful capabilities that are not directly available in the nodes that Model Studio provides. The SAS Code node provides ultimate flexibility in what you can incorporate into your pipelines. You can execute SAS procedures and write SAS DATA steps to create customized scoring code, conditionally process data, or manipulate existing data sets. The Code node is also useful for building predictive models that are not supported by existing nodes, for formatting SAS output, for defining table and plot views in the user interface, and for modifying variable metadata. This node can create output plots and tables that show up as results just as they do for other nodes, and data tables that are produced by a successful Code node execution can be used by subsequent nodes in a pipeline.

Tip: Because the Code node can be used to run any SAS code you want, it defaults to being used for preprocessing. However, you can move it to be a Supervised Learning node by clicking the node menu (⋮) and selecting **Move**. This allows it to automatically connect to the Model Comparison node.

MODEL COMPARISON

All supervised learning models automatically feed into a Model Comparison node to allow for side-by-side assessment and comparison of all candidate models within your pipeline.

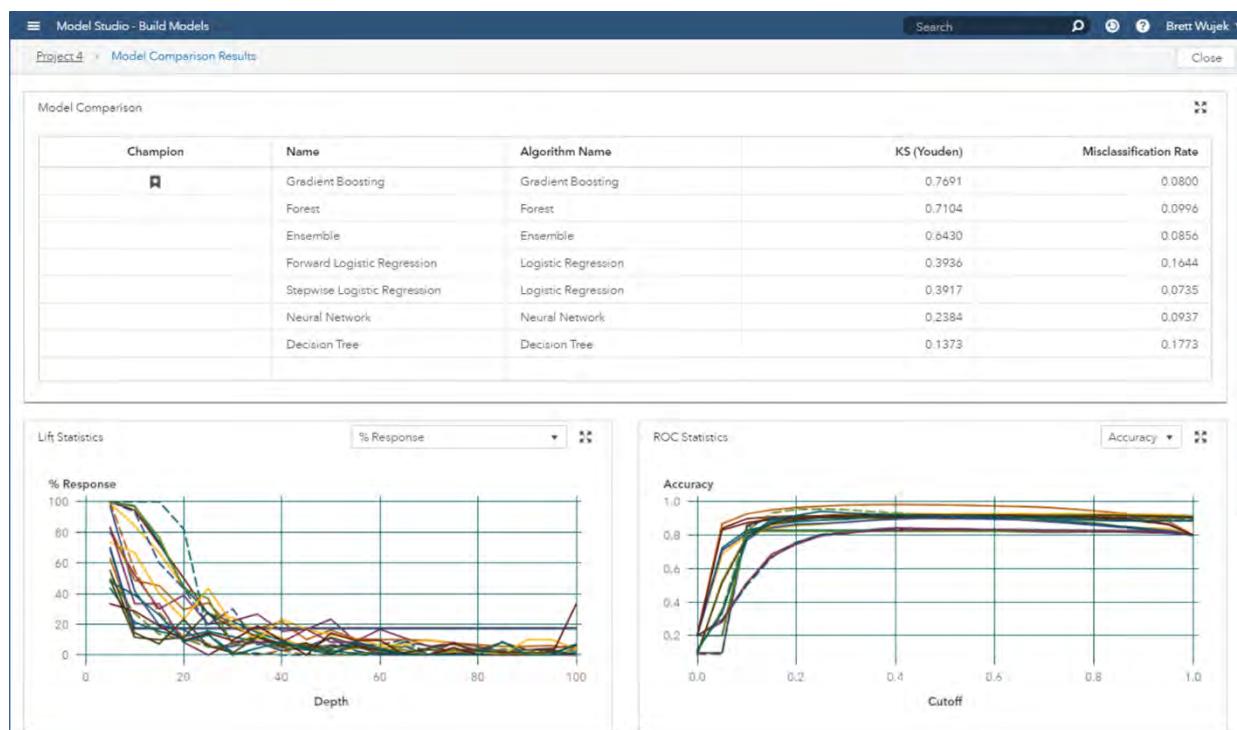


Figure 21. Comparing Models Generated by a Pipeline in Model Studio

The first table you see in the results of the Model Comparison node contains high-level information for each of your candidate models. The first column indicates the champion model that is automatically selected for you on the basis of your requested model selection statistic. You can configure this model selection statistic in the properties of the Model Comparison node; by default, it is based on the Kolmogorov-Smirnov statistic for class targets and the average squared error for interval targets. You can also select the partition that is used for champion selection; by default, the validation partition is used if it is available. In addition, the results of this node contain a comprehensive comparison of fit statistics and standard assessment plots across each of your candidate models, including lift, gain, %Response, and ROC, to name a few for a class target.

Tip: You can change the default statistic, partition, and cutoff that are used for comparing models and selecting a champion in the user settings for Model Studio. To set the default values of these options as desired, select <your user name> → **Settings** in the upper right.

The Model Comparison node automatically identifies and flags a champion model for each of the *pipelines* in your project, but your ultimate goal is to identify an overall champion for your project. This can be done on the **Pipeline Comparison** tab of Model Studio.

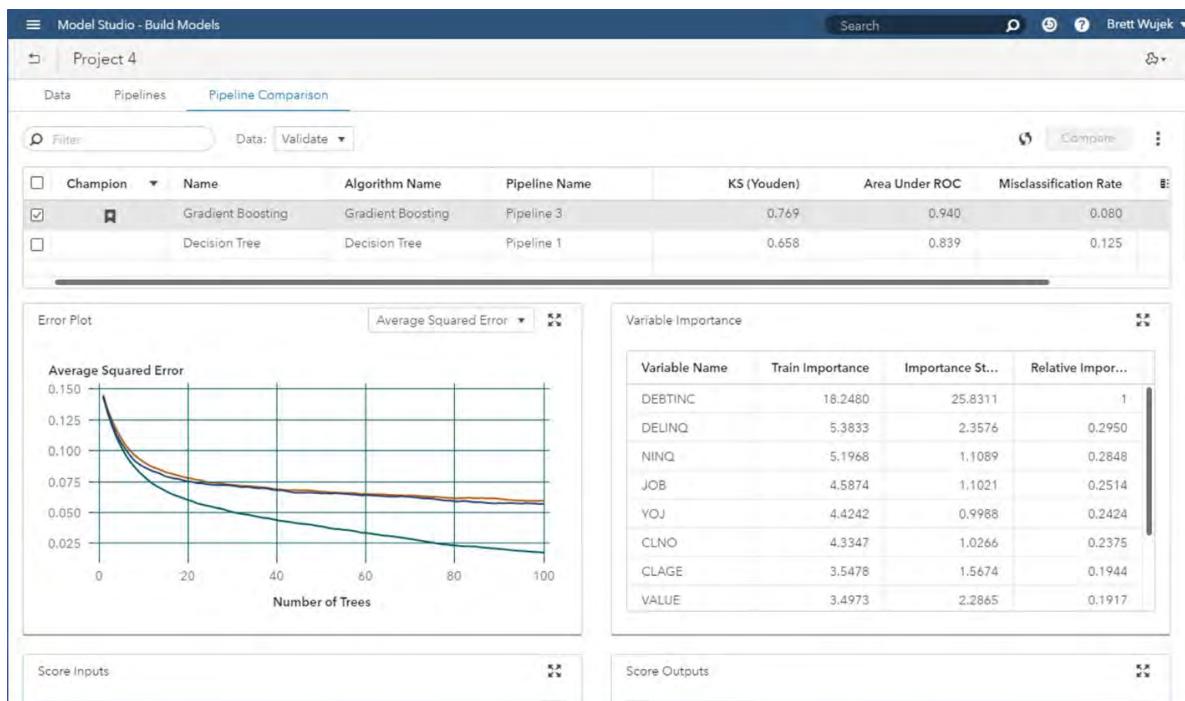


Figure 22. Comparing Models from All Pipelines in the Project in Model Studio

The look and feel of this tab is very similar to the results of the Model Comparison node (shown in Figure 21). The table at the top of this tab contains information for each of the candidate models in the project. In this case, your candidate models are the champion models that were identified by the Model Comparison node in each of your pipelines. As is done in the Model Comparison node, an overall project champion is identified from your candidate models based on your specified model selection criterion.

Tip: You can also manually add challenger models (models that you want to be evaluated and compared to determine a champion) from any pipeline into the Pipeline Comparison table by selecting the associated node menu (⋮) and selecting **Add challenger model**.

As you select individual models within this table, model-specific results are displayed in addition to assessment tables and plots, scoring code, and a list of required inputs and generated output variables for your model. If you select multiple models within this table, you can generate a side-by-side comparison of these models by clicking **Compare**.

There are times when you might want to include models that are generated outside of Model Studio to consider in the determination of your overall project champion. A perfect example would be when you want to compare models you have created in SAS® Enterprise Miner™ to new models that are generated in Model Studio. The score code for these external models can be easily imported into the Pipeline Comparison table by selecting **Import score code** from the actions menu (⋮), as shown in Figure 23.

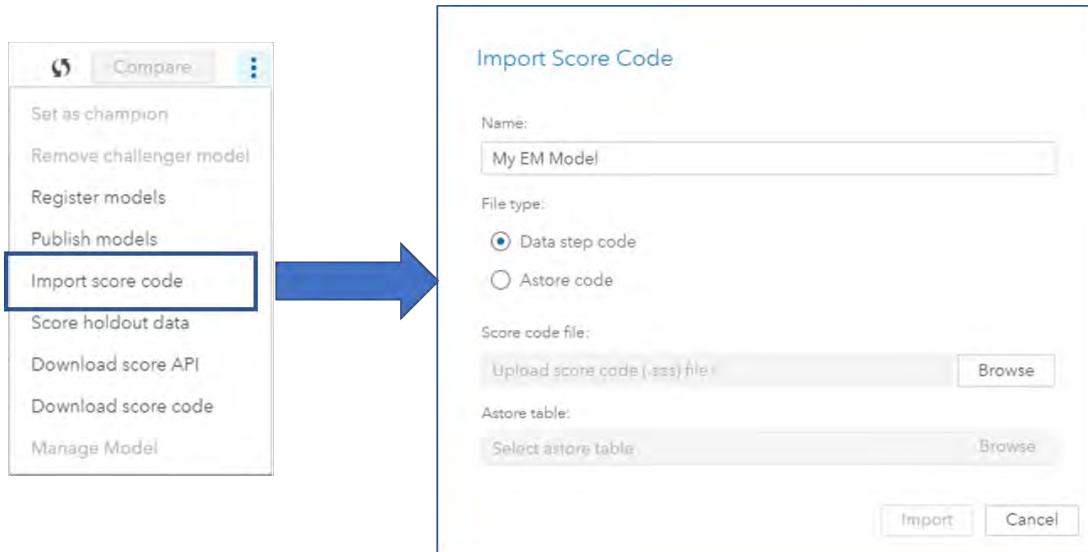


Figure 23. Importing Model Score Code in Model Studio

After this scoring code has been imported, it is applied to the project data, each of the partitions is assessed, and fit statistics are calculated. This model is now also included in the table of project candidates and is considered like any other candidate model during the overall project champion selection. No associated pipeline is created to visualize this external model.

In order to truly assess how well a model will generalize and perform, the model needs to be applied to a separate holdout table—a table that was not used in the creation of the models or in the validation for automatic selection of the champion. You can choose this table by selecting **Score holdout data** from the actions menu, as shown in Figure 23. This option opens a data browser that enables you to drill into available libraries and choose the appropriate CAS table to be used as a holdout. After the table is chosen, the score code for all project candidate models is applied to this holdout data. After the score code has been applied, the table and all assessment plots and tables for each model are updated to include values for the holdout sample.

Tip: Although Model Studio automatically identifies a champion model for your project based on a model selection statistic, you can manually override the champion by highlighting the desired model and selecting **Set as champion** from the action menu (⋮).

MODEL DEPLOYMENT AND MANAGEMENT

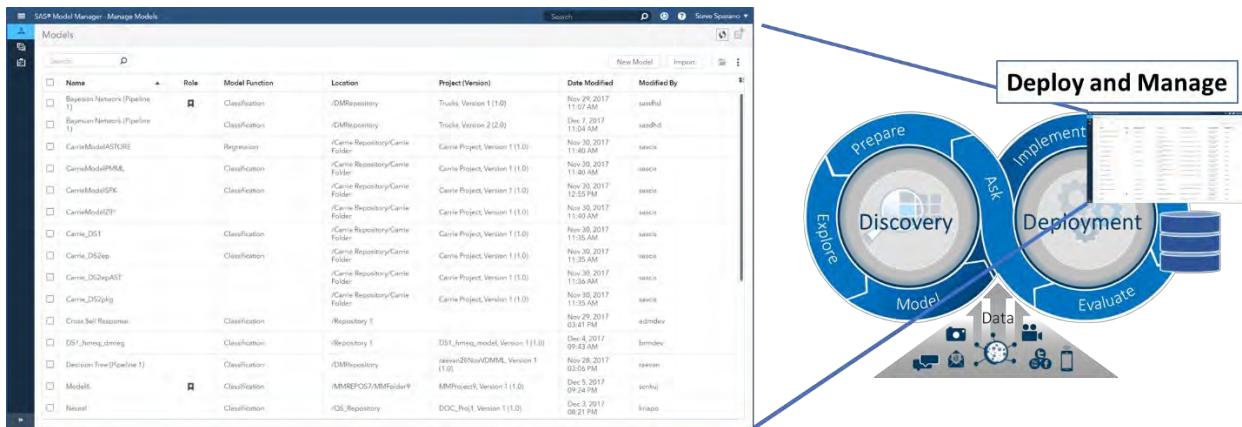


Figure 24. Managing Models Deployed by SAS Visual Data Mining and Machine Learning

Now that you have built candidate models and determined a champion, the next step in the analytics life cycle is to deploy and manage your models to aid in making effective business decisions. Deployment capabilities are surfaced through several options on the action menu (⋮) on the **Pipeline Comparison** tab in Model Studio.

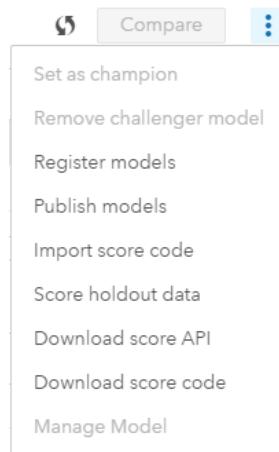


Figure 25. Actions Available for Using Models Created in Model Studio

A powerful way of deploying your models to your production environment is to publish them to destinations that support executing them to score new data. If you select **Publish models**, you are prompted to select a destination, which can be CAS, Hadoop, or Teradata, depending on the existence of a license for the corresponding SAS® Scoring Accelerator. As part of the publishing process, the model is translated into scoring code that can be seamlessly executed within the production environment.

Tip: Before you can publish models to a caslib for CAS, Hadoop, or Teradata, that caslib must be specified as a publishing destination by an administrator.

Alternately, you can execute models directly via a scoring web service call by selecting **Download score API**. This action provides you with the code that uses a model to score data by issuing a REST call, wrapped in SAS, Python, or simply the REST call itself. You can also download the SAS score code itself to execute in a SAS environment; for models in the form of analytic stores, the associated astore file is saved to the Models caslib.

Although using your models to score new data is a fundamental goal, it is just as important to maintain a level of organization for the models and a measure of control over them. If you have a license for SAS®

Model Manager, you can also register your models into a common model repository. This repository supports version control of your models, and it enables you to monitor stability and performance over time, and to update and retrain models when necessary, as described in Clingroth (2018). Proper management of the models that are used in your production environment is a necessary step to ensure that they are adequately and accurately addressing your business problem. With this management in place, you can confidently incorporate these models with other business rules into your decision process and operational workflows by using SAS® Decision Manager (if licensed) to automate analytic-based decision making.

CONCLUSION

SAS Visual Data Mining and Machine Learning is not just a set of algorithms. It is not a specific user interface or programming module. Rather, it is a comprehensive, fully integrated assembly of capabilities and interfaces that accommodate the entire analytics life cycle, enabling you to smoothly navigate from data to decisions. Built on the foundation of SAS Viya, it exploits the full power of your available computing resources with the latest innovations in in-memory analytics for efficient execution on distributed data. It offers full flexibility with support for interactive, programmatic, or automated approaches to applying analytics to your data. The collaborative environment enables users of all levels of expertise, from programmer to business analyst, to participate in the process of using modern machine learning techniques to turn data into real business value.



Figure 26. End-to-End Navigation of the Analytics Life Cycle with SAS Visual Data Mining and Machine Learning

APPENDIX: CASE STUDY

This case study illustrates how you can use SAS Visual Data Mining and Machine Learning on SAS Viya to solve a modern business problem. It takes you through the process of preparing data, interactively exploring data and building models, building automated modeling pipelines, and deploying models—all within one environment.

Imagine that you are a data scientist at a telecommunications company and you are charged with the task of identifying the most likely customers to drop their service within the next year. You have data sets that represent account information and usage patterns.

Tip: *The data and other artifacts that are associated with this case study can be found at https://github.com/sassoftware/sas-viya-machine-learning/tree/master/case_studies/telecom*

PREPARE DATA

Two tables have been loaded into memory:

- TELECOM_ACCOUNTS, which contains information about account attributes
- TELECOM_USAGE, which contains usage attributes.

These two tables are joined together by customer_id in the Data Studio interface. If there were additional data tables to join, they could also be joined in here.

The screenshot shows the SAS Data Studio interface with the title 'SAS® Data Studio - Prepare Data'. On the left, a sidebar lists various data quality transforms: Column Transforms, Change case, Convert column, Rename, Remove, Split, Trim whitespace, Custom Transforms, Calculated column, Code, and Data Quality Transforms. Under 'Data Quality Transforms', 'Change case' is highlighted. The main workspace is titled 'Join - Step 1 of 4' and shows a 'Join Tables' dialog. It lists 'Table 1 (T1)' as 'TELECOM_ACCOUNTS' and 'Table 2 (T2)' as 'TELECOM_USAGE'. A join condition is defined: 'T1.Customer_ID' is equated to 'T2.Customer_ID'. Below the dialog, a preview of the joined data is shown in a table with columns: customer, region, region_lat, first_name, last_name, item_id, and age. The preview data includes rows for customers like Carmen, Logan, and Salem across different regions and ages.

Figure 27. Join Multiple Tables Using SAS Visual Data Mining and Machine Learning on SAS Viya

An important aspect of data preparation is to identify potential areas for data cleansing. For example, you might have data such as city and state that are in mixed case and have varying spellings and abbreviations. Figure 28 shows a visual profile of the data. As you can see, the variable customer_state contains values of mixed case. The predictive models would treat values such as Massachusetts and MASSACHUSETTS as different values. You can automatically standardize the values by using the **Change case** transform.

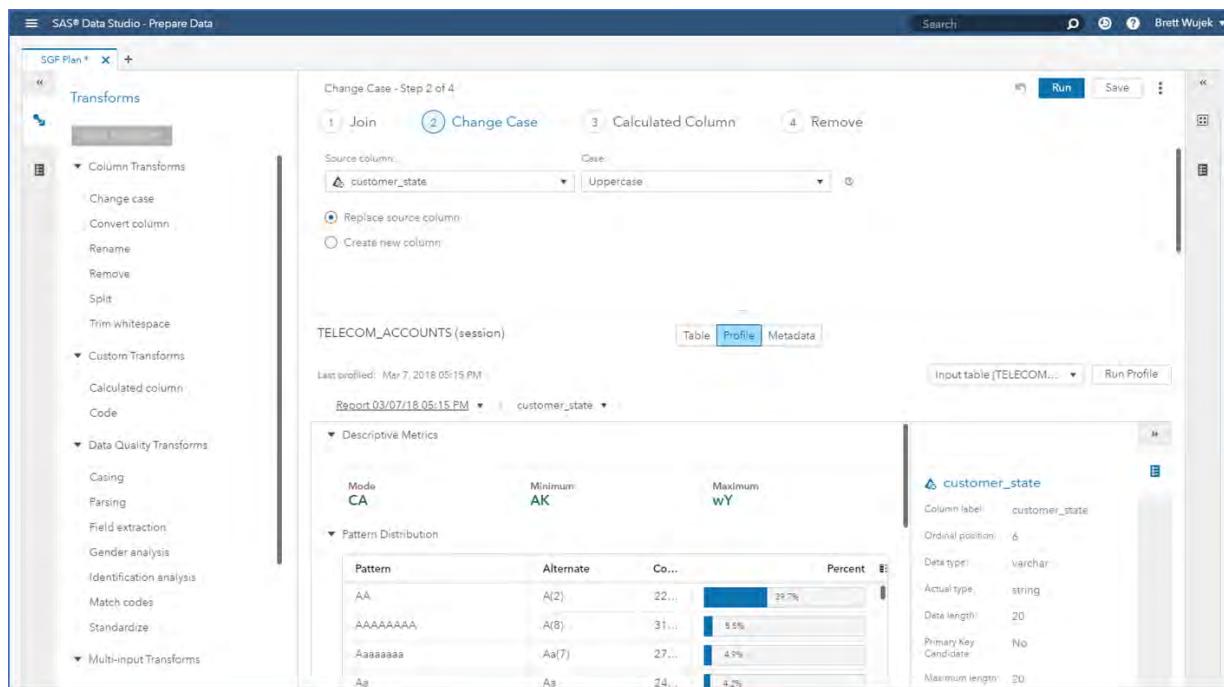


Figure 28. Generate Variable Profiles and Identify and Fix Data Quality Issues

The column that contains the churn information contains two numeric values: 1 and 0, which represent YES and NO respectively. Use the **Calculated column** transform to add custom SAS code that executes the IFC SAS function to transform the values to YES and NO.

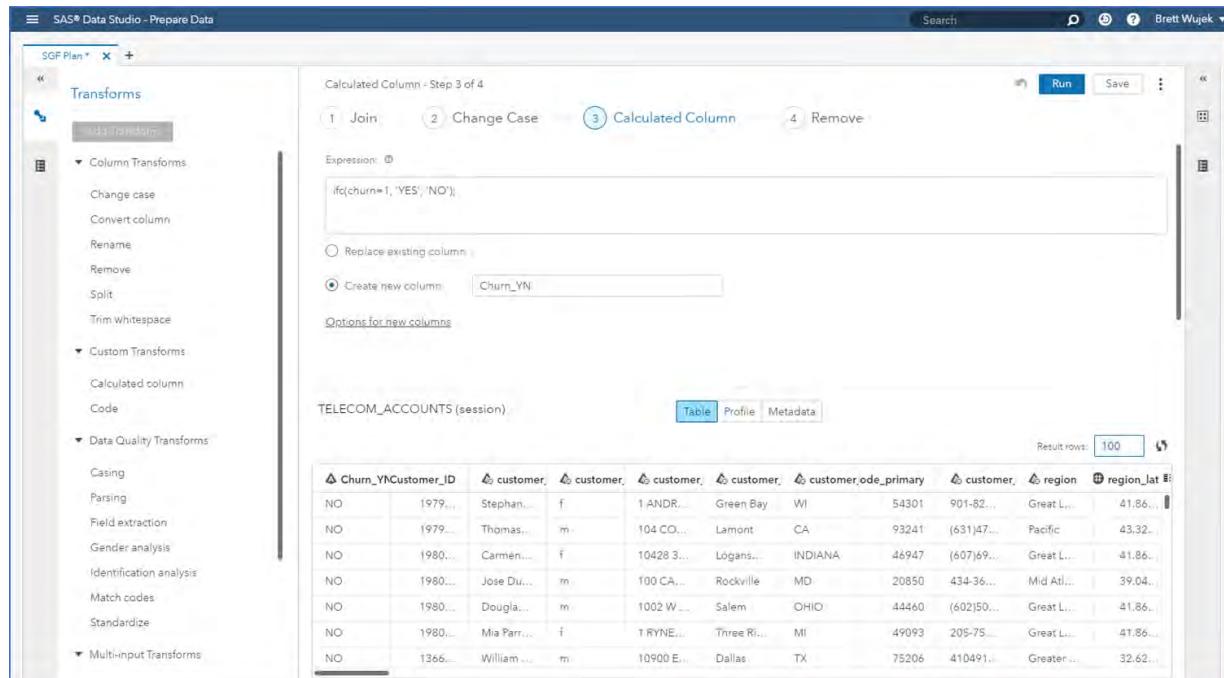


Figure 29. Create Calculated Columns by Using Custom SAS Programming Functions

Now that you have joined your data tables together, standardized the values of customer_state, and created the target column, you can save this data preparation “plan” for future use. This analytic base table can be shared with other users in the system for collaborating and running analyses in parallel.

Sharing the table reduces the need to manually copy data, thus preserving a single center of truth. With preliminary data preparation complete, you can now explore and visualize the data.

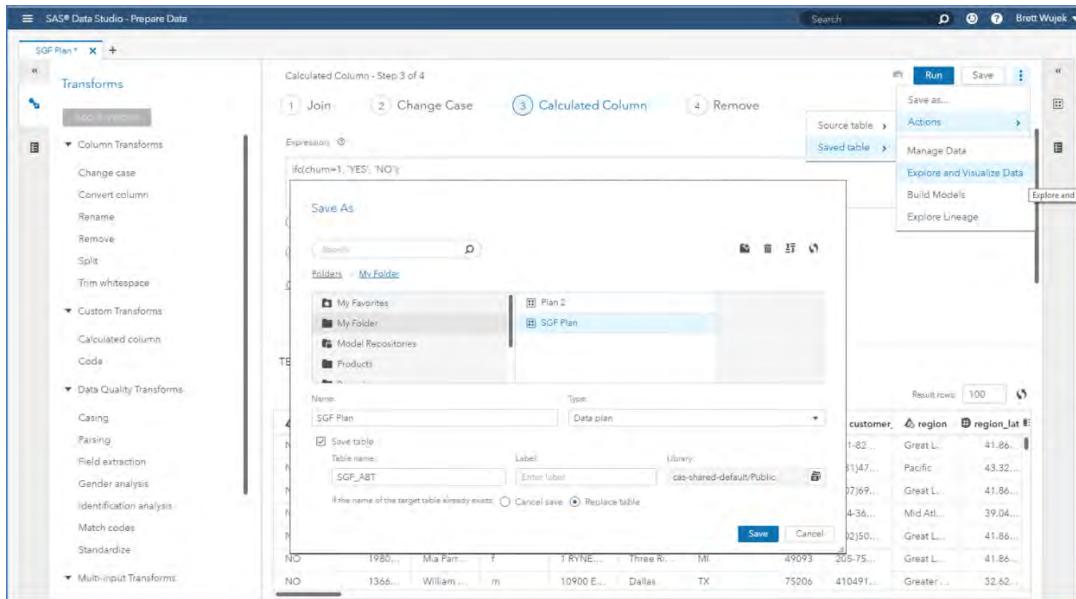


Figure 30. Save Data and Plan for Additional Analyses

EXPLORE AND VISUALIZE

Once you click **Explore and Visualize Data**, another interface automatically appears and presents the data that you just prepared. The data table has not been copied; rather the application launched and pointed to the same data table in memory.

Figure 31 shows a bar chart that was created to present the distribution of the target variable. Approximately 4.16% of the customers churned within a year. A correlation matrix of the continuous attributes in the data was also created. The darker shaded cells indicate a strong correlation between the variables.

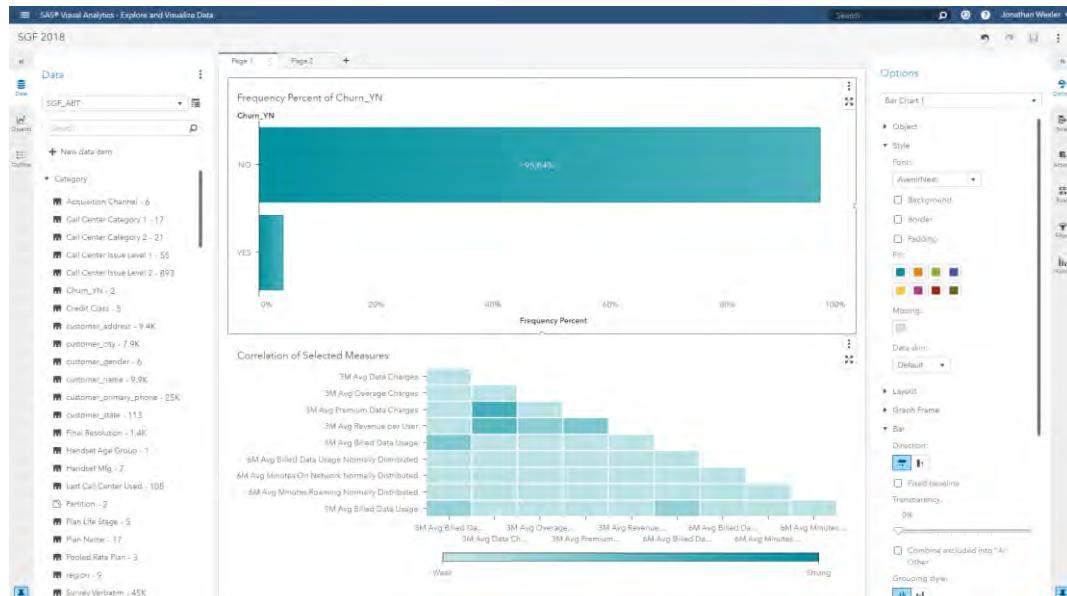


Figure 31. Interactively Explore and Visualize Patterns in Data Using SAS Visual Data Mining and Machine Learning on SAS Viya

To identify the customers most likely to churn, you can build a gradient boosting machine model and partition the data to hold out 30% for validation. This partition information is represented in the data as a new column.

As you can see in Figure 32, the Validation Misclassification is 0.0230, meaning 2.3% of your validation set observations were not correctly classified. The default settings of the models were used, so this gives you a good baseline to start. You can see that Times Suspended Last 6M and Total Voice Charges are the two most important factors in identifying customer churn. The variable importance column is interactive, so you can remove columns that do not add value to the model.

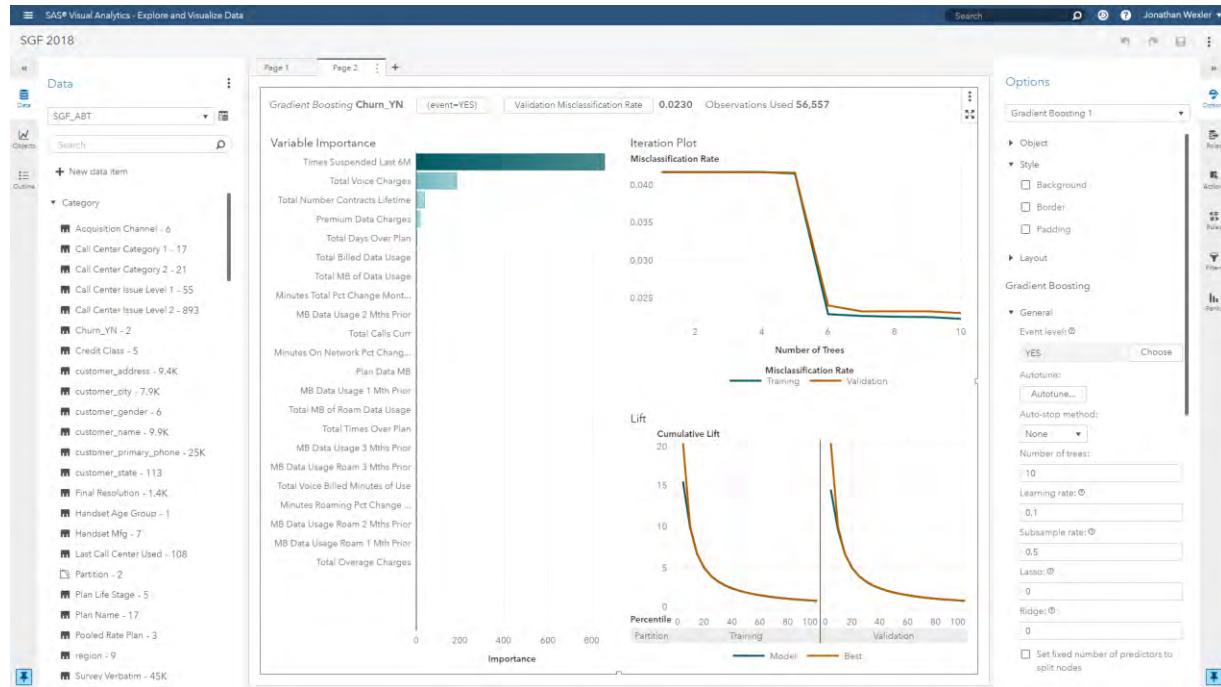


Figure 32. Interactively Build and Tune a Gradient Boosting Machine Model

You could spend time manually adjusting algorithm settings (hyperparameters) for this model to minimize misclassification. Instead, you can invoke this process automatically by using autotuning (click **Autotune**), which uses optimization methods to intelligently search for the optimal set of hyperparameters for your model. In Figure 33, you can see that using autotuning improved Validation Misclassification to 0.0223. In this case, the best set of hyperparameter values were four auto-stop iterations, 150 trees, a learning rate of 0.421, a subsample rate of 1, and lasso and ridge regularization parameters of 5.34 and 6.84, respectively. Manually experimenting to find these values by trial-and-error would be infeasible.

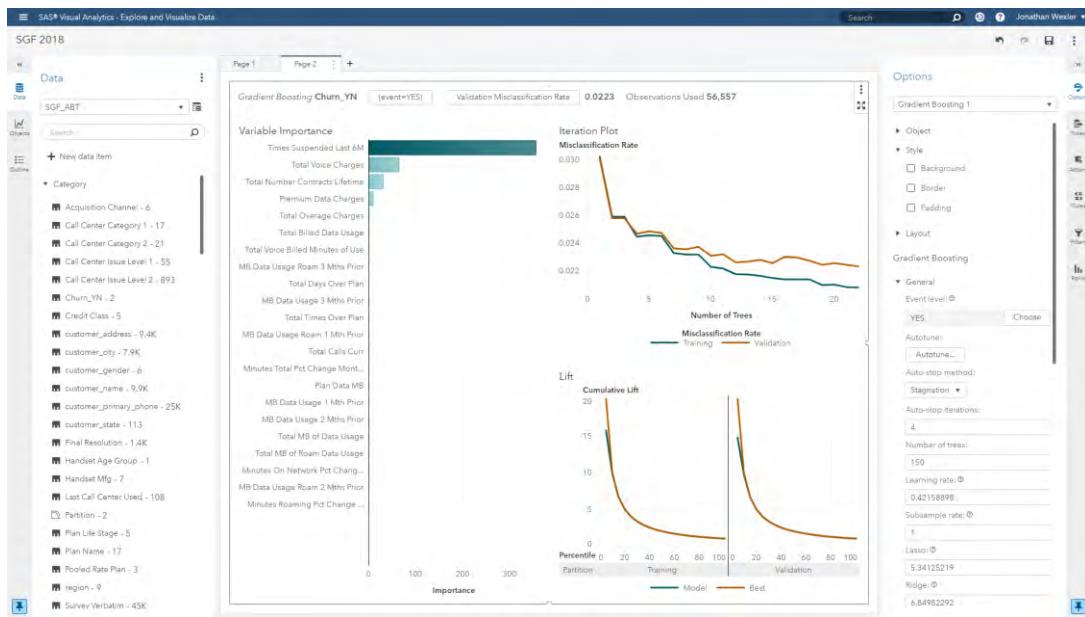


Figure 33. Use Autotuning to Find the Optimal Set of Hyperparameters for Your Model

BUILD MODELS

You could spend additional time tuning your gradient boosting machine and building more models such as neural networks and forests. However, at this point you might want to capture your interactive process and preserve it for reuse. When you right-click on the model results and select **Create pipeline**, any interactive steps that were executed for data preparation and the generated gradient boosting machine model are represented as a pipeline, as shown in Figure 34. If you had created a data transformation, that would be represented in the Interactive Data Prep node. Note that the score code for the gradient boosting machine that was generated interactively is automatically incorporated as a modeling (Supervised Learning) node in the pipeline. This node can be used for assessment and comparison, but the model properties cannot be changed for retraining in this software release (SAS Visual Data Mining and Machine Learning 8.2).

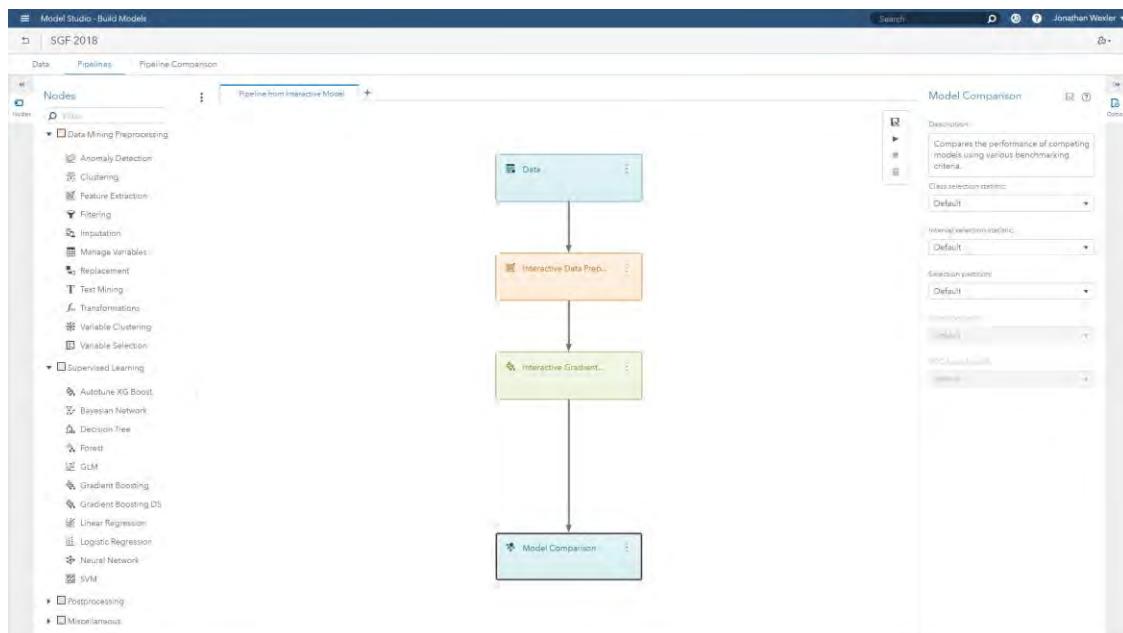


Figure 34. Generate Automated Pipeline for Additional Feature Engineering and Modeling

In Figure 35, this pipeline is enhanced by imputing missing values, adding tree-based binning interval transformations, and feeding the transformed data into nodes to generate a stepwise logistic regression model and an autotuned gradient boosting machine model.

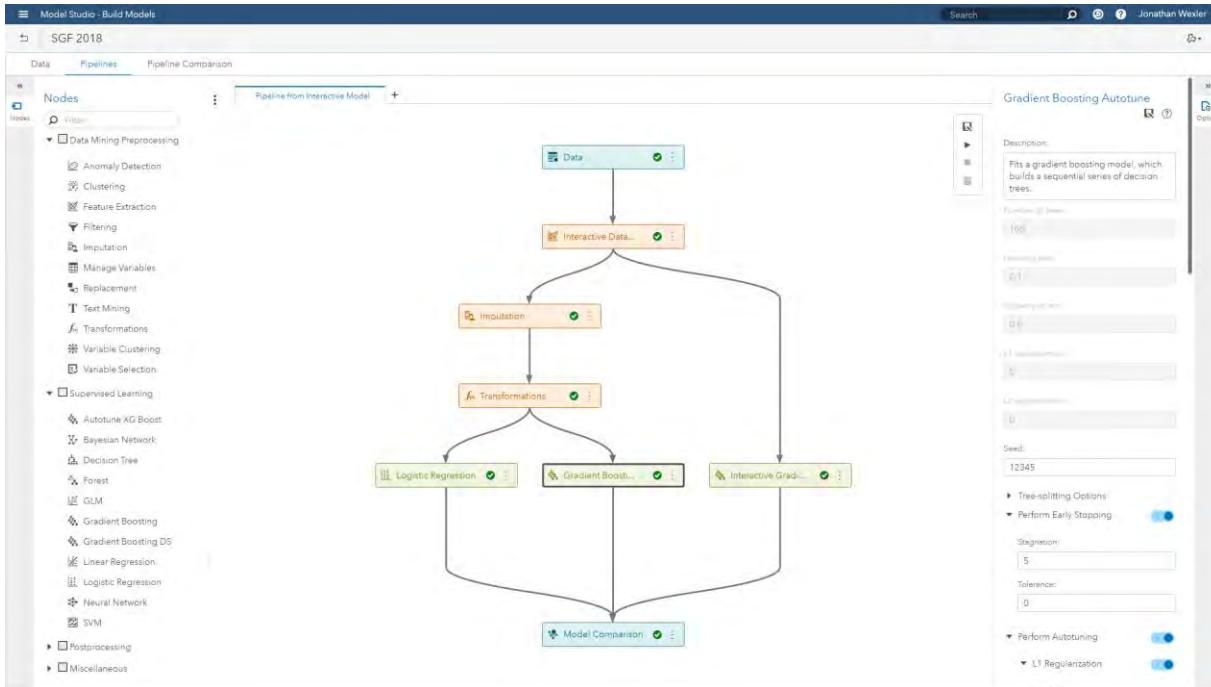


Figure 35. Enhance Pipeline with Additional Analytic Methods and Automatically Choose the Best Model

The pipeline automatically chose the best performing model, based on lowest validation misclassification rate. The gradient boosting autotune model had the lowest misclassification rate at 0.0217.

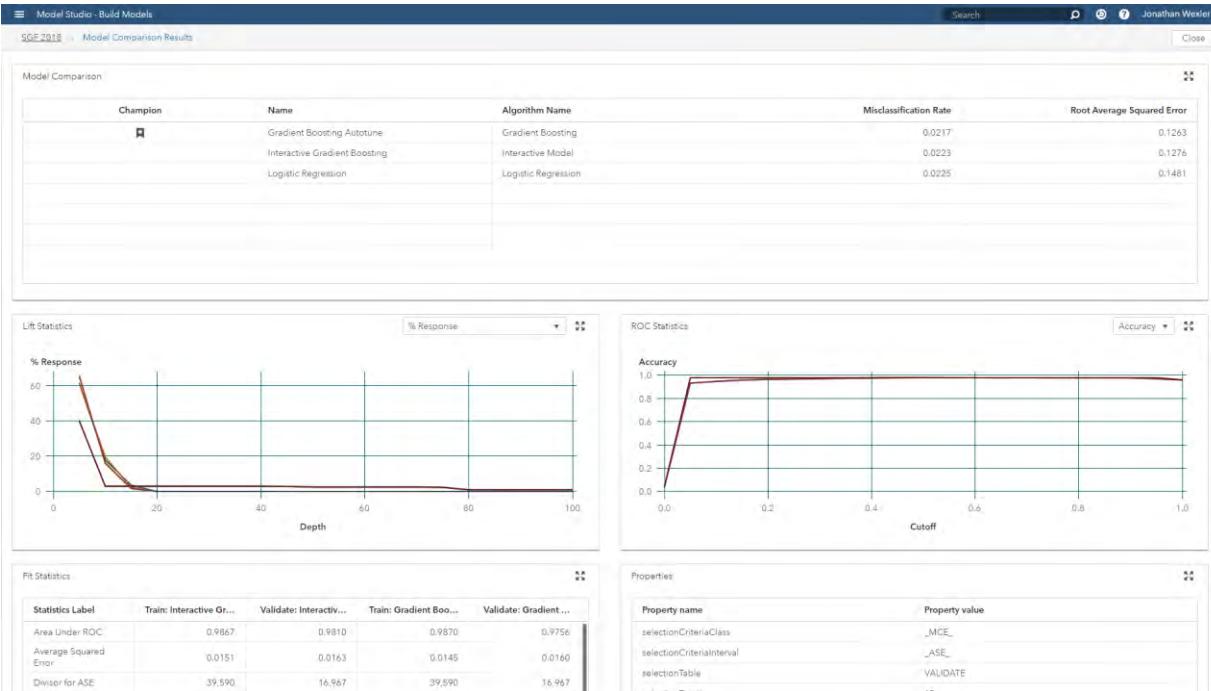
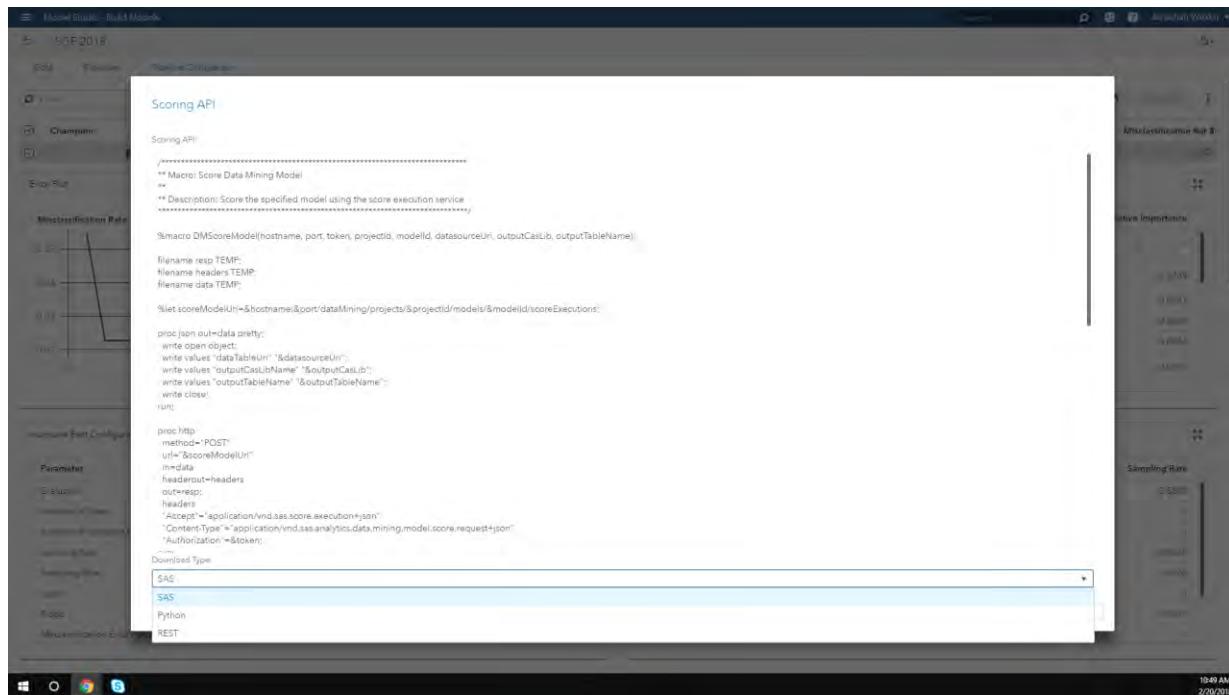


Figure 36. Automatically Select Champion Model Based on Assessment Statistics

After you have trained several models, including a selected champion model and any potential challenger models, you have complete flexibility with respect to model deployment. Figure 37 shows the code that is generated to invoke a SAS scoring API. For example, if you had a web application, you could use this code to call back into SAS to automatically score new records. You could just as easily use the Python or REST APIs to deploy your models. If your production data that needed to be scored resided in Hadoop or some relational database, you could also publish these pipelines automatically with just two mouse clicks, depending on SAS licensing. The pipelines would be embedded in the production systems, with no manual recoding.

Note that when the time comes to update your models, you can also retrain this entire pipeline by using a batch retrain API that the application generates.



```

Scoring API
-----
** Macro: Score Data Mining Model
**
** Description: Score the specified model using the score execution service
-----
%macro DMScoreModel(hostname, port, token, projectId, modelId, datasourceUri, outputCasLib, outputTableName);
filename resp TEMP;
filename headers TEMP;
filename data TEMP;

%let scoreModelUrl=&hostname||&port||dataMining/projects/&projectId/models/&modelId/scoreExecution;

proc json out=resp pretty;
  write open object;
  write values "dataTableUri" "&datasourceUri";
  write values "outputCasLibName" "&outputCasLib";
  write values "outputTableName" "&outputTableName";
  write close;
run;

proc http;
method=POST;
url=&scoreModelUrl;
inData;
headers;
  accept="application/vnd.sas.score.execution+json";
  "Content-Type"="application/vnd.sas.analytics.data.mining.model.score.request+json";
  "Authorization"=&token;
...
Download Type
SAS
SAS
Python
REST

```

Figure 37. Deploy Pipelines to Production Using In-Database/Hadoop Publishing or Scoring APIs

REFERENCES

- Clingroth, G. “Introducing SAS Model Manager 15.1 for SAS Viya.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc.
- Koch, P., Wujek, B., Golovidov, O., and Gardner, S. (2017). “Automated Hyperparameter Tuning for Effective Machine Learning.” *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings17/SAS0514-2017.pdf>.
- Koch, P., Wujek, B., and Golovidov, O. (2018). “Managing the Expense of Hyperparameter Autotuning.” *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc.
- Wexler, J., Haller, S., and Myneni, R. 2017. “An Overview of SAS Visual Data Mining and Machine Learning on SAS Viya.” *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available <https://support.sas.com/resources/papers/proceedings17/SAS1492-2017.pdf>.
- Wujek, B., Hall, P., and Güneş, F. (2016). “Best Practices in Machine Learning Applications.” *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available <https://support.sas.com/resources/papers/proceedings16/SAS2360-2016.pdf>.

ACKNOWLEDGMENTS

The authors would like to thank Anne Baxter for her contributions to this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Brett Wujek
SAS Institute Inc.
brett.wujek@sas.com

Susan Haller
SAS Institute Inc.
susan.haller@sas.com

Jonathan Wexler
SAS Institute Inc.
jonathan.wexler@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Managing the Expense of Hyperparameter Autotuning

Patrick Koch, Brett Wujek, and Oleg Golovidov

SAS Institute Inc.

ABSTRACT

Determining the best values of machine learning algorithm hyperparameters for a specific data set can be a difficult and computationally expensive challenge. The recently released AUTOTUNE statement and **autotune** action set in SAS® Visual Data Mining and Machine Learning automatically tune hyperparameters of modeling algorithms by using a parallel local search optimization framework to ease the challenges and expense of hyperparameter optimization. This implementation allows multiple hyperparameter configurations to be evaluated concurrently, even when data and model training must be distributed across computing resources because of the size of the data set.

With the ability to both distribute the training process and parallelize the tuning process, one challenge then becomes how to allocate the computing resources for the most efficient autotuning process. The best number of worker nodes for training a single model might not lead to the best resource usage for autotuning. To further reduce autotuning expense, early stopping of long-running hyperparameter configurations that have stagnated can free up resources for additional configurations. For big data, when the model training process is especially expensive, subsampling the data for training and validation can also reduce the tuning expense. This paper discusses the trade-offs that are associated with each of these performance-enhancing measures and demonstrates tuning results and efficiency gains for each.

INTRODUCTION

Machine learning predictive modeling algorithms are governed by “hyperparameters” that have no clear defaults agreeable to a wide range of applications. The *depth* of a decision tree, *number of trees* in a forest or a gradient boosting tree model, *number of hidden layers* and *neurons in each layer* in a neural network, and *degree of regularization* to prevent overfitting are a few examples of quantities that must be prescribed. Not only do ideal settings for the hyperparameters dictate the performance of the training process, but more importantly they govern the quality of the resulting predictive models. Tuning hyperparameter values is a critical aspect of the model training process and is considered to be a best practice for a successful machine learning application (Wujek, Hall, and Güneş 2016). Manual hyperparameter adjustment and rough grid search approaches to tuning are recently being traded for automated intelligent search strategies. Random search has been shown to perform better than grid search, particularly when the number of influential hyperparameters is low (Bergstra and Bengio 2012). With increased dimensionality of the hyperparameter space (that is, as more hyperparameters require tuning), a manual tuning process becomes much more difficult even for experts, grid searches become more coarse and less practical because they grow exponentially with dimensionality, and random search requires many more samples to identify candidate models with improved accuracy. As a result, numerical optimization strategies for hyperparameter tuning have become more popular for intelligent search of complex hyperparameter spaces (Bergstra et al. 2011; Eggensperger et al. 2013). Optimization for hyperparameter tuning typically can very quickly reduce, by several percentage points, the model error that is produced by default settings of these hyperparameters. Parallel tuning allows exploration of more configurations, further refining hyperparameter values and leading to additional improvement.

SAS® Visual Data Mining and Machine Learning, described in Wexler, Haller, and Myneni (2017), provides a hyperparameter autotuning capability that is built on SAS® local search optimization (LSO). SAS LSO is a hybrid derivative-free optimization framework that operates on the SAS® Viya® distributed analytics execution engine to overcome the challenges and expense of hyperparameter optimization. This implementation of autotuning, detailed in Koch et al. (2017), is available in the TREESPLIT, FOREST, GRADBOOST, NNET, SVMACHINE, and FACTMAC procedures by using the AUTOTUNE statement. Statement options define tunable hyperparameters, default ranges, user overrides, and validation schemes to avoid overfitting. The procedures that incorporate the AUTOTUNE statement invoke corresponding actions in the **autotune** action set. These actions (**tuneDecisionTree**, **tuneForest**,

tuneGradientBoostTree, **tuneNeuralNet**, **tuneSvm**, and **tuneFactMac**) can also be executed directly on SAS Viya.

As shown in Figure 1, the LSO framework consists of an extendable suite of search methods that are driven by a hybrid solver manager that controls concurrent execution of search methods. Objective evaluations (different model configurations in this case) are distributed across multiple worker nodes in a compute cluster and coordinated in a feedback loop that supplies data from running search methods. As illustrated in Figure 2, the autotuning capability in SAS Visual Data Mining and Machine Learning uses a default hybrid search strategy that begins with a Latin hypercube sample (LHS), which provides a more uniform sample of the hyperparameter space than a grid or random search provides. The best configurations from the LHS are then used to seed a genetic algorithm (GA), which crosses and mutates the best samples in an iterative process to generate a new population of model configurations for each iteration. The strengths of this approach include handling continuous, integer, and categorical variables; handling nonsmooth, discontinuous spaces; and ease of parallelizing the search strategy. All of these challenges are prevalent and critical in hyperparameter tuning problems. Alternate search methods include a single Latin hypercube sample, a purely random sample, and a Bayesian search method. It is important to note here that the LHS or random samples can be evaluated in parallel and that the GA population or Bayesian samples at each iteration can be evaluated in parallel.

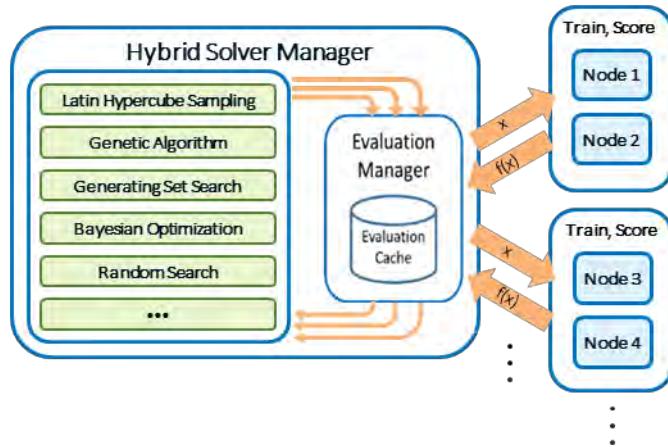


Figure 1. Autotuning with Local Search Optimization: Parallel Hybrid Derivative-Free Optimization Strategy

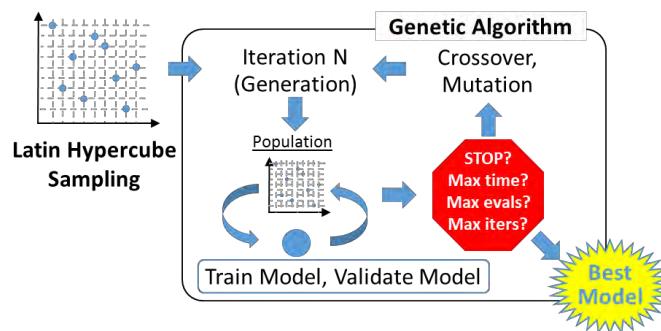


Figure 2. Default Autotuning Process in SAS Visual Data Mining and Machine Learning

An automated, parallelized, intelligent search strategy can benefit both novice and expert machine learning algorithm users. Challenges still exist, however, particularly related to the expense of hyperparameter tuning. Primary contributors to the expense of hyperparameter tuning are discussed in the next section. Options to manage these expenses within the SAS autotuning implementation are then presented in the following section, with examples that demonstrate expense management trade-offs. Best-practice recommendations are offered in conclusion.

HYPERPARAMETER TUNING EXPENSES

Even when a compute cluster is used both to distribute large data sets for model training and to concurrently evaluate multiple model hyperparameter configurations in parallel, hyperparameter tuning is a computationally expensive process. Often many configurations must be evaluated in pursuit of a high-quality model. One challenge becomes deciding how to best allocate compute resources. The LSO-driven hyperparameter process in Figure 1 depicts the use of two worker nodes for each model training, with multiple models trained in parallel. Are two worker nodes per model training necessary? Ideal? Figure 3 illustrates different possibilities for hyperparameter tuning on a compute cluster that has 8 worker nodes. If all 8 worker nodes are used for each model training, the training time might be reduced, but the tuning process becomes sequential. A sample of 8 hyperparameter configurations could all be evaluated in parallel, with one worker evaluating each configuration, without overloading the cluster, but the size of the data set might demand more workers to train a model. Perhaps allocating four workers for model training and training two models in parallel, or allocating two workers for model training and training four models in parallel, is appropriate. The best worker allocation for hyperparameter tuning depends on the training expense, the savings observed with parallel tuning, the size of the cluster, and to some degree the hyperparameter ranges (which dictate how complex the models become). The best number of worker nodes for a single model training might not lead to the best resource usage for autotuning.

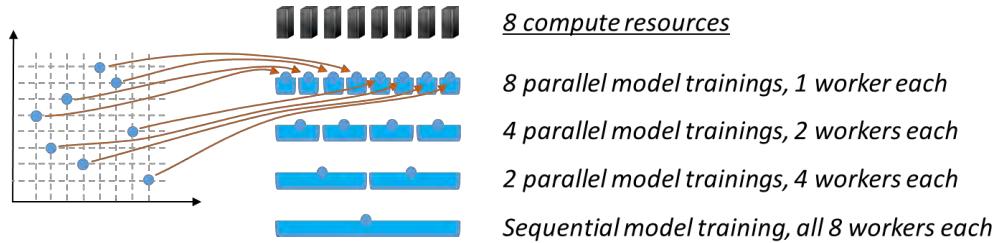
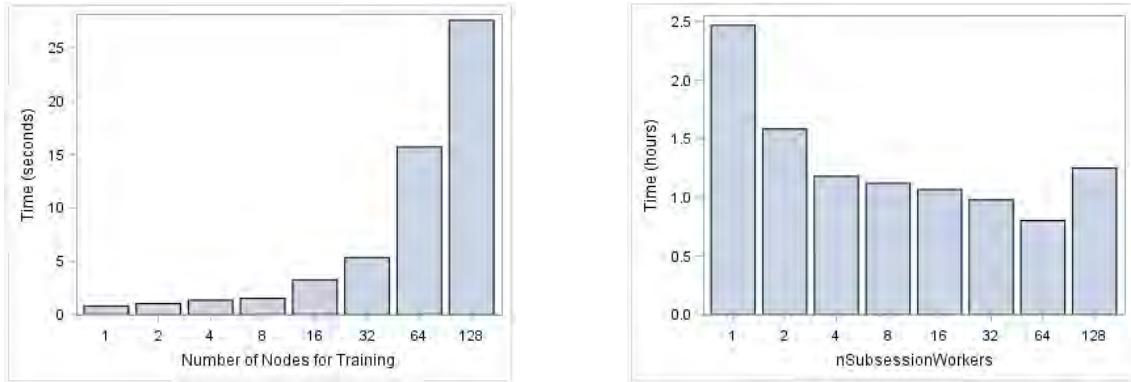


Figure 3. Use of Compute Resources for Tuning Many Models

One extreme case for resource allocation that might not be immediately obvious is that of a small data set. As shown in Figure 4(a), the training expense actually increases when the number of worker nodes is increased. The expense of communication between nodes adds to the training expense, which is most efficient when all the data are on a single node for small data sets. In this case, available workers should be used for parallel tuning of different hyperparameter configurations for increased efficiency of tuning—with as many models trained concurrently as possible or desired. However, as data sets grow, both in length and width (many inputs to a model can have a larger effect on training expense than many observations), training time is reduced by increasing the number of worker nodes, up to a certain number of workers. When the number of workers passes some threshold, the communication cost again leads to an increase in training time, as shown in Figure 4(b). In this case, resource allocation is not straightforward. Even though a single model training is most efficient on 64 workers, tuning might not be most efficient if every hyperparameter configuration to be evaluated uses 64 workers. If the cluster contains 128 workers, only two models could be evaluated in parallel during tuning without overloading the cluster. Furthermore, different hyperparameter configurations vary in expense; fewer hidden layers and neurons in a neural network or fewer trees in a forest are more efficient to train. Most importantly, however, the training expense shown in Figure 4(b) with 64 workers is not half the expense with 32 workers. In fact, it is slightly more than half the expense of training on two workers. If each model to be trained uses two workers, the cluster of 128 workers would accommodate 64 models trained in parallel during tuning rather than two models in parallel if 64 workers are used for each model training. More hyperparameter configurations can be evaluated in the same amount of time, or less time is needed for evaluating a specific number of hyperparameter configurations.



(a) 150 observations, 5 columns

(b) 50,000 observations, 3073 columns

Figure 4. Training Expense for Data Sets of Different Size

Although training multiple hyperparameter configurations in parallel can significantly reduce the expense of tuning, there are additional contributing factors to consider. First, much of the expense of hyperparameter tuning is spent on model configurations that are not only worse than the current best model (or even the default model), but are often quite bad. As shown in Figure 5, although the best configuration from a Latin hypercube sampling of model candidates has a 6% misclassification rate, most have more than 10% error, many have more than 20% error, and quite a few have worse than 40% error. The use of an intelligent search strategy that is designed to *learn* over multiple iterations, such as the default strategy in the LSO framework described previously, helps reduce the number of bad configurations over time. Still, significant expense can be incurred to complete the training process for model configurations that might have stagnated (ceased to make meaningful improvement) before the training process has completed. Training all model configurations to completion (beyond the point of stagnation) is especially costly for large data sets and complex model configurations (which can delay the completion of an iteration, because all candidate models within an iteration must complete training before the next iteration can start). Ideally, stagnated configurations are identified and the time spent training these models is reduced. Furthermore, the expense of model training, compounded in model tuning, is also obviously tied to the size of the data set. This is clear in Figure 4 where the smaller data set training time is measured in seconds and the larger data set training time is measured in hours; training on the entire data set during tuning might not be necessary and might not be the most efficient approach.

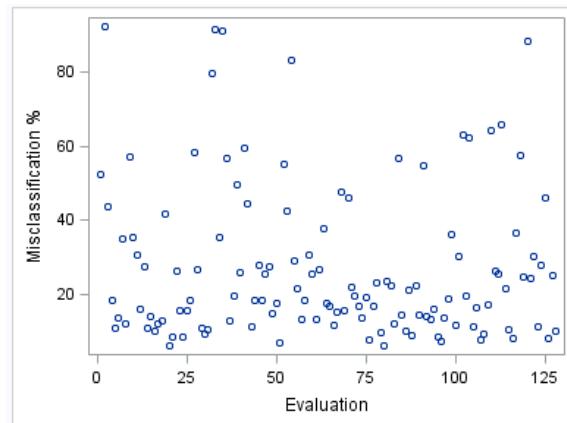


Figure 5. Latin Hypercube Sample of Candidate Models—Many Bad Configurations

EFFICIENT AUTOTUNING ON SAS VIYA

Given the various factors that contribute to the expense of hyperparameter tuning and the trade-offs to be made based on the data set and compute resources at hand, flexibility is required for an efficient autotuning solution. SAS Viya actions are executed within a “session” that uses one or more worker nodes. The autotuning implementation running on SAS Viya creates additional “subsessions,” which are managed from the parent session, in order to facilitate parallel training of different model configurations by isolating each alternate configuration within a separate subsession with its own set of worker nodes. SAS Viya automatically handles the data management for execution in subsessions. The autotuning implementation enables control of the expense of hyperparameter tuning through the following:

- resource allocation of worker nodes for training versus tuning
- early stopping of stagnated models
- subsampling large data sets for faster training times

Table 1 shows the procedure options and corresponding action parameters that correspond to these controls. They are discussed further in this section, with results from demonstration problems provided to illustrate their effectiveness and associated trade-offs. All these controls are configured with defaults that are designed to reduce the anticipated expense of the autotuning process based on the data set size and the available compute resources, and they can be adjusted further to trade off the tuning expense and the accuracy of models that are generated. For simplicity in the following text, when a control can be specified either in a procedure option or in a corresponding action parameter, the control is presented only by the procedure option name and syntax (in all capital letters).

Procedure Options		Action Parameters
Resource Allocation	NSUBSESSIONWORKERS	nSubsessionWorkers
	NPARALLEL	nParallel
Early Stopping	EARLYSTOP, STAGNATION, VALIDATION	earlyStop
Subsampling	PARTITION	trainFraction, validateFraction

Table 1. Autotuning Efficiency Controls

Before each of these controls for managing the expense of autotuning is discussed in more detail, an example is provided here to familiarize you with the associated syntax, for both the GRADBOOST procedure and for the **autotune.tuneGradientBoostTree** action. All procedures that include the AUTOTUNE statement—TREESPLIT, FOREST, GRADBOOST, NNET, SVMACHINE, and FACTMAC—include the NSUBSESSIONWORKERS and NPARALLEL options, which, in addition to the POPSIZE option, can be used to adjust the resource allocation for tuning. The example here uses a small data set, so the number of workers per subsession (NSUBSESSIONWORKERS) for model training is set at 1 (which is the default for this data set size) and the number of parallel model configurations (NPARALLEL) is adjusted to match a cluster size of 30 workers. The population size (POPSIZE) is also increased to make full use of the compute resources; it is set to 31 (with the default search method, the best model from the previous iteration is included but does not need to be retrained). The procedure or action results in a maximum of 150 configurations being evaluated with five iterations (the default). Early stopping (EARLYSTOP) is activated, directing the modeling algorithms to terminate training if they stagnate for four consecutive iterations. With the procedures, the PARTITION statement can be used to implement subsampling of training data (not necessary in this small data set case, but shown for illustration). With a 0.2 fraction defined for TEST and 0.3 for VALIDATE, half the data (0.5) will be used for training. If the TEST fraction is not defined, the fraction used for training would be 0.7.

```

cas mysess sessopts=(nworkers=1);

libname mycaslib sasioca casref=mysess;

data mycaslib.dimagecr;
  set sampsio.dimagecr;
run;

proc gradboost data=mycaslib.dimagecr outmodel=mycaslib.mymodel
  earlystop(stagnation=4);
  partition fraction(test=0.20 validate=0.30);
  target good_bad / level=nominal;
  input checking duration history amount savings employed installp
    marital coapp resident property age other housing existcr job
    depends telephon foreign / level=interval;
  input purpose / level=nominal;
  autotune nsubsessionworkers=1 nparallel=30 popsize=31
    evalhistory=all;
run;

```

The number of parallel evaluations and worker nodes for each evaluation is reported in a log note when this code runs; if the NPARALLEL option was not specified in the procedure call, this note indicates the automated decision for the resource allocation.

NOTE: Autotune number of parallel evaluations is set to 30, each using 1 worker nodes.

After execution, if the model that contains the best found hyperparameter configuration terminated early as a result of stagnation, a log note indicates how many trees were used in the final model (which will be less than the value selected by the tuner during tuning).

NOTE: Due to early stopping, the actual final number of trees used in the model (19) is less than the Autotune selected 'best' value (75).

The following **tuneGradientBoostTree** action call is equivalent to the PROC GRADBOOST call. In this action call, all the parameters for managing the tuning expense are provided in the *tunerOptions* parameter, except for the *earlyStop* parameter, which is available only in the **tuneGradientBoostTree** and **tuneNeuralNet** actions.

```

proc cas noqueue;
  autotune.tuneGradientBoostTree /
    tunerOptions={
      nSubsessionWorkers=1, nParallel=30, popsize=31,
      trainFraction=0.50, validateFraction=0.30, loglevel=3
    },
    earlyStop=true,
    trainOptions={
      table={name='dimagecr'},
      inputs={'checking', 'duration', 'history', 'amount',
        'savings', 'employed', 'installp', 'marital'}
    }
  ;

```

```

        'coapp', 'resident', 'property', 'age', 'other',
        'housing', 'existcr', 'job', 'depends',
        'telephon', 'foreign', 'purpose'},
    target='good_bad',
    nominals={'purpose', 'good_bad'},
    casout={name='dmagecr_gbt_model', replace=true}
}
;
run;
quit;

```

An additional log note is provided with the action execution in this case because early stopping was not explicitly included. By default, the **tuneGradientBoostTree** action includes early stopping with *stagnation*=4. If the EARLYSTOP option is omitted from the PROC GRADBOOST syntax, the use of the AUTOTUNE statement will still activate early stopping with STAGNATION=4 and the following log note would also be included.

NOTE: Automatic early stopping is activated with STAGNATION=4; set EARLYSTOP=false to deactivate.

Table 2 lists the data sets used for the tuning efficiency studies that are presented in this section. These data sets range from tall and relatively narrow to short and very wide. They are listed in increasing order of number of values in the data set. The width of the data set (the number of attributes) has a significant impact on training, and hence on tuning expense. A more detailed description of each data set is provided in Appendix A.

	# Observations	# Attributes	# Classes	# Values
Covtype	581,012	54	7	31,955,660
MNIST	60,000	718	10	43,140,000
Bank	1,060,038	54	2	57,242,052
CIFAR-10	50,000	3072	10	153,650,000

Table 2. Benchmark Data Sets Summary

RESOURCE ALLOCATION AND NUMBER OF PARALLEL MODELS

As illustrated in Figure 4, it is clear that distributed training is not only unnecessary for small data sets, it is inefficient because of the cost of communication between worker nodes. The best allocation of resources for hyperparameter tuning with relatively small data sets would be to use a single worker node for each hyperparameter configuration, allowing all worker nodes to be used for parallel evaluation of different model configurations during the tuning process. With larger data sets, the cost of each individual model training must be weighed against the cost of the tuning process overall, while considering the maximum potential number of parallel evaluations (based on the search method), the complexity of the models being investigated, and the time budgeted. Thus, the “best” resource allocation is affected by many factors, including the following:

- size of the data set used for model training

- the search method and its configuration: population size for GA or Bayesian search methods, or sample size for random or LHS search methods
- number of workers available to the server

For example, as illustrated in Figure 6, if the population for each iteration contains 16 new configurations to be evaluated, a cluster of 128 workers would support 8 workers per model configuration, even though it might not be notably more efficient to train with 8 workers compared to 4 workers. Alternately, by using only 2 workers per model configuration, although each model training might take a little longer, the number of configurations to evaluate could be increased to 64, with all configurations still being trained in parallel. The choice depends on preference—reduced time with faster individual model training, or more candidate model configurations in the same amount of time.

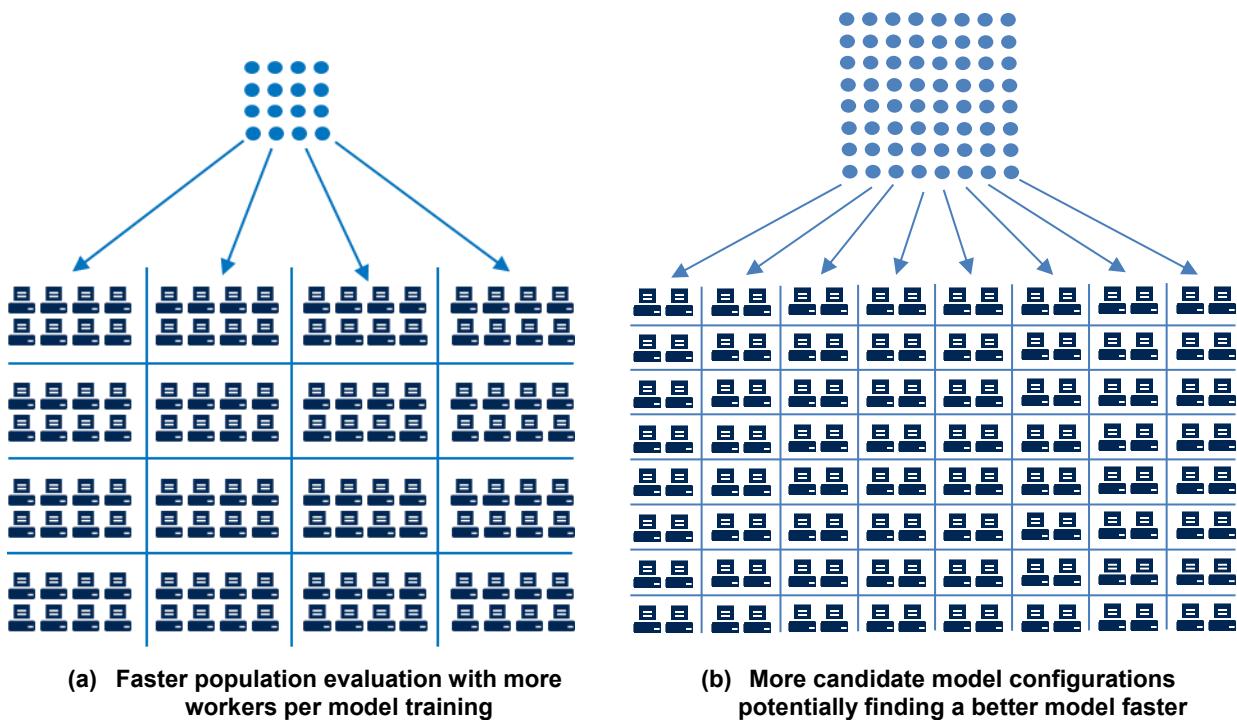


Figure 6. Resource Allocation for an Individual Population Evaluation

The default population size for autotuning in SAS Visual Data Mining and Machine Learning is set conservatively at 10 model configurations per iteration, for a default of five iterations—a maximum of 50 model configurations. The number of workers to use for each model training and the number of parallel evaluations are controlled by the NSUBSESSIONWORKERS and NPARALLEL options, respectively. Default values of these options are determined based on the data set size and the cluster size. First, the number of workers to use for each model training is determined. If the NSUBSESSIONWORKERS option is not specified, the number of workers is determined based on the size of the data set:

$$\text{NSUBSESSIONWORKERS} = 1 + \frac{n\text{DataRows} * n\text{DataColumns}}{50 \text{ million}}$$

The default number of workers in each subsession (each used for one model training) is set at one node per 50 million values, aggressively favoring allocation of resources to parallel tuning.

After the number of workers to use for each model training has been determined, the number of model trainings that can execute in parallel can be calculated. First, the number of *potential* parallel evaluations is determined based on the search method: one less than the population size for the GA search method (accounting for the best point carried over from the previous iteration), the population size for the Bayesian search method, or the sampling size for the random or LHS search methods. The *actual* number of parallel evaluations is then limited by the server configuration. In single-machine mode, if the number of *potential* parallel evaluations is greater than 4, it is limited to 4. This limit can be overridden up to a value of 32 by specifying the NPARALLEL option. In distributed mode, the upper limit for the number of parallel evaluations is calculated as W/n , where W is the number of workers connected to the server and n is the number of workers used for each model training. This limit can be overridden by up to a factor of 2 by specifying the NPARALLEL option, with a maximum value of $2W/n$. This resource allocation process is summarized in Figure 7.

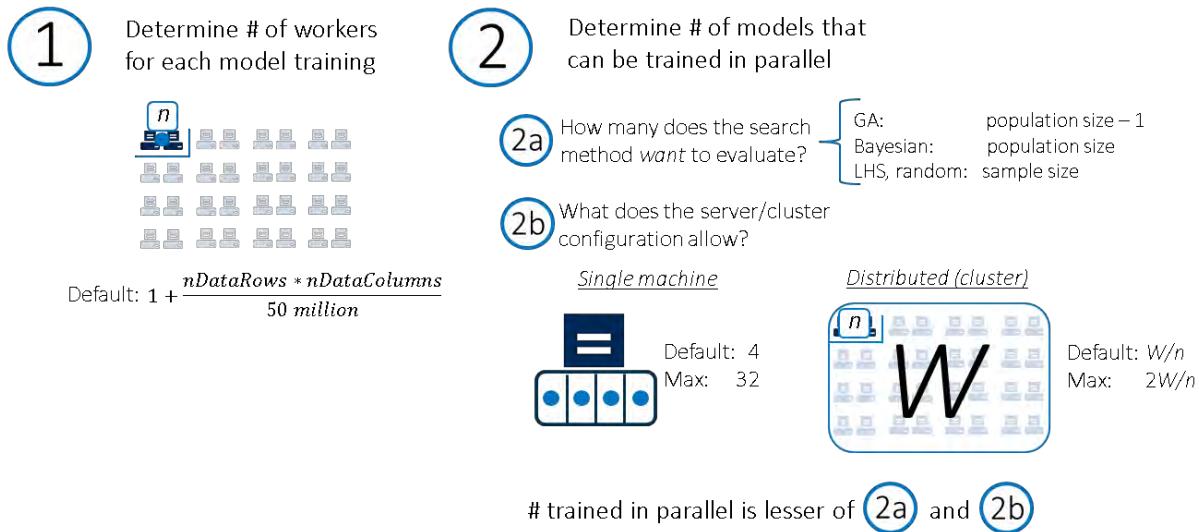


Figure 7. Process for Determining Worker Allocation (Training versus Tuning)

As an example, consider a data set that has 1.5 million observations and 50 columns, for a total of 75 million values. Based on the preceding equation for NSUBSESSIONWORKERS, two workers will be assigned to each subsession by default. With the default tuning search method, NPARALLEL will be set to 9 based on the default population size of 10; thus, a total of 18 worker nodes in the cluster will be required. If the cluster contains only 16 worker nodes, NPARALLEL will be reduced to 8 by default, or can be overridden to as many as 16 (overloading the workers). If 38 workers are available, either the population size can be increased to 20 (19 new models to train at each iteration) to make use of all the workers (with 2 workers per parallel subsession) or the number of workers per subsession could be increased to 4 for faster model training if the default maximum number of configurations is desired. What should be avoided is keeping population size at 10 when NPARALLEL is reduced to 8 (16-worker cluster). In this case, eight models will be submitted in parallel, and the remaining model will be submitted when one of the first eight models finishes and frees up the subsession workers, with the other seven subsessions being idle. With roughly equal training times (which is not usually the case), each iteration then requires two batches, or roughly the cost of two model trainings, rather than the cost of a single training (when all models in the population are evaluated in parallel). Population size must be carefully considered and adjusted manually when necessary.

Case Study Results

For each of the data sets listed in Table 2, numerous studies were run, using different allocations of compute resources for individual model training (NSUBSESSIONWORKERS) and parallel tuning (NPARALLEL). Results for each data set are shown in Figure 8 through Figure 11. On the left in each pair

of results plots, plot (a) shows the time for a single model training for a number of resource allocation configurations—a single model is trained using 1, 2, 4, 8, 16, or 32 worker nodes. On the right in each pair of results plots, plot (b) shows the time for the default autotuning process—population size of 10 with a maximum of five iterations—for the same number of worker nodes allocated to each model configuration that is trained. The plots also display the number of parallel models. With a cluster of 32 available worker nodes, all models in an iteration can be evaluated in parallel if the number of worker nodes for each model is limited to one or two workers. (Recall that with a population size of 10, nine new models are generated and trained at each iteration because the one best model is carried forward after each iteration.) With four workers for model training, eight models can be tuned in parallel, using all 32 workers. When eight or 16 workers are used per model, four or two models, respectively, can be tuned in parallel. If all 32 workers are used for training, the tuning process is sequential: one model is trained at a time.

Figure 8 through Figure 11 clearly show not only that allocating more workers for training does not necessarily continue to increase the training efficiency, but also that the most efficient number of workers for model training is not the most efficient configuration for model tuning. This difference is a result of the efficiency gains from parallel tuning, which requires worker nodes to be available for allocation to different model configurations. For each case study data set, the default autotuning resource allocation is also indicated. The default number of worker nodes used for each model configuration, based on data set sizes, is set to one worker for the CoverType and MNIST data sets, two workers for the Bank data set, and four workers for the CIFAR-10 data set. For all but the CIFAR-10 data set, it is clear that the default resource allocation is not the most efficient. The number of workers nodes for each configuration is chosen to allow more models to be trained in parallel. With the CoverType and MNIST data sets, it would be possible to increase the population size to train up to 32 models in parallel in each iteration. For the Bank data set, 16 models could be trained in parallel in each iteration. Alternately, if only the default number of total configurations is desired, the number of workers used for each model can be increased to reduce the tuning time, as shown in the plots. The parallel speed up with the default tuning process is also reported to indicate that even if the default resource allocation is not the most efficient configuration, it is more efficient than sequential tuning.

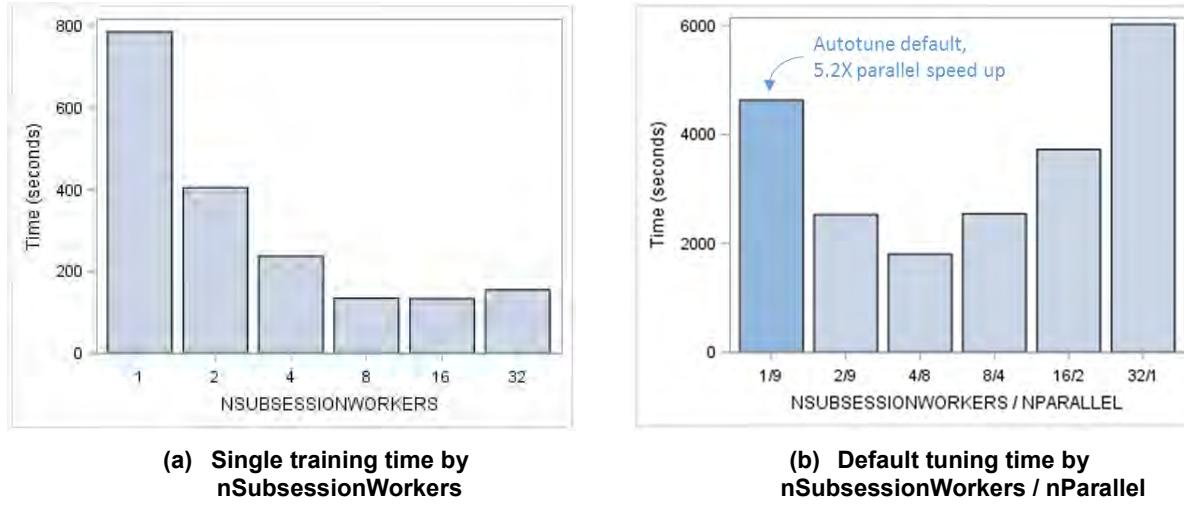


Figure 8. Resource Allocation Comparisons, CoverType Data Set

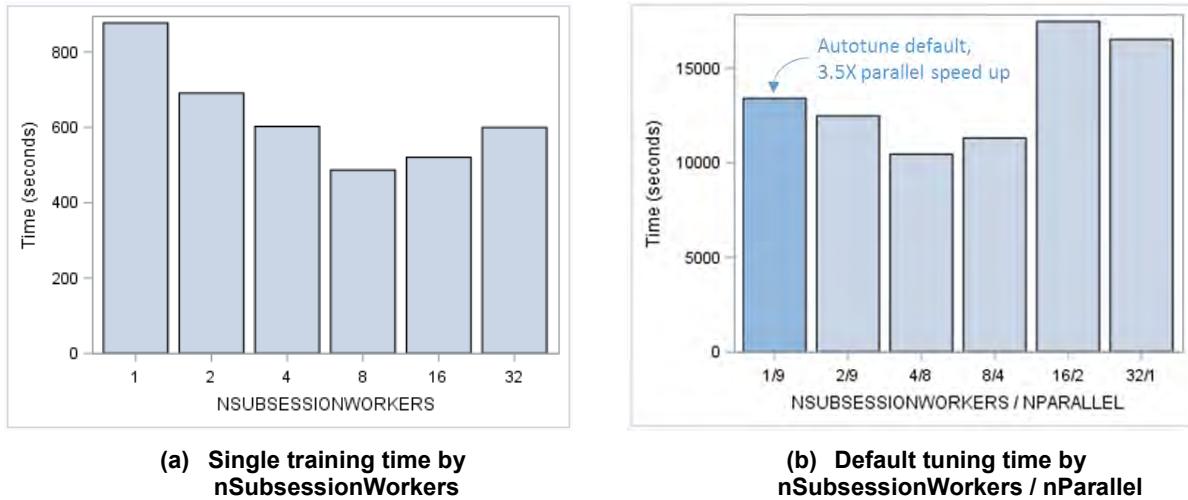


Figure 9. Resource Allocation Comparisons, MNIST Data Set

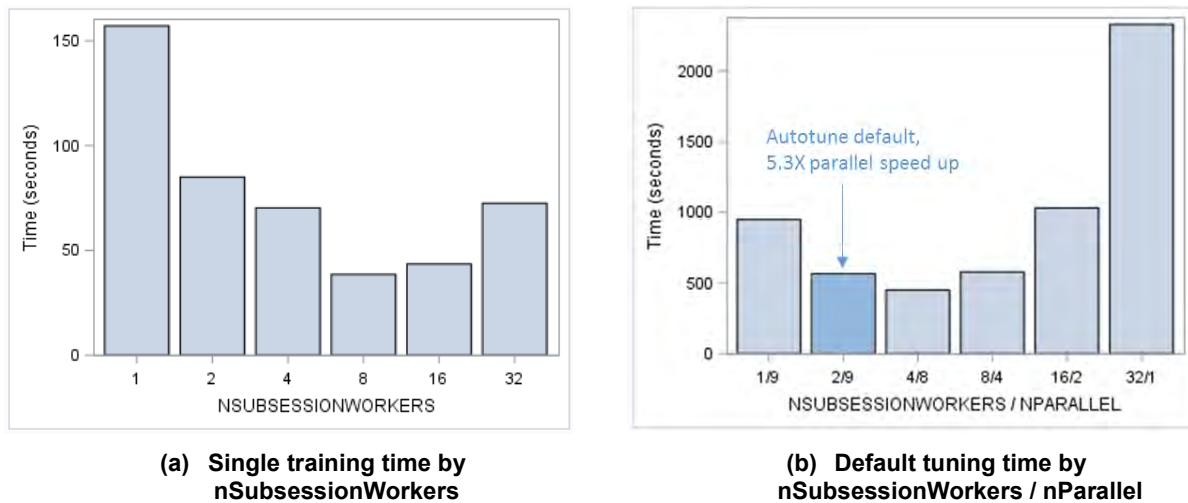


Figure 10. Resource Allocation Comparisons, Bank Data Set

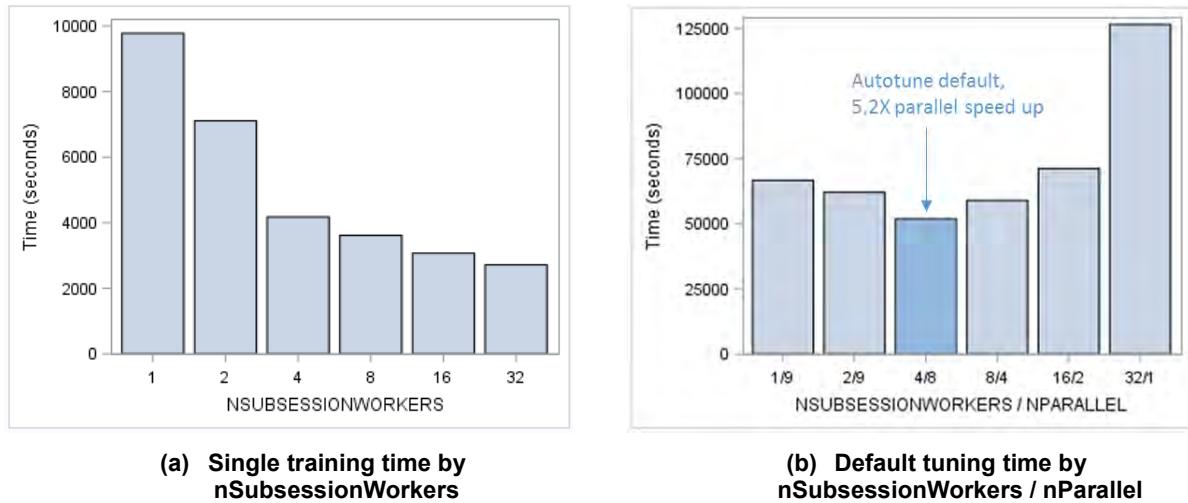


Figure 11. Resource Allocation Comparisons, CIFAR-10 Data Set

Figure 12 illustrates two alternate autotuning configurations that use the Bank data set. Using four workers per model configuration results in the fastest training time for this data set. However, with 32 workers available, only eight models can be tuned in parallel if four workers are used for each model. With the default population size of 10, leading to nine new models generated in each iteration (one carried forward from the previous iteration), there will be two batches for each iteration: eight in the first batch, and then the ninth will be evaluated as soon as one of the subsessions is available. In this case, it is more effective to set the population size to nine, resulting in eight new configurations at each iteration, all evaluated in a single batch. The number of iterations can then be increased and roughly the same number of configurations will be evaluated in less total time—six iterations with a single batch each (six submission batches in total) versus five iterations with two batches each (10 submission batches in total). These results are shown in Figure 12(a). In Figure 12(b), the default autotuning configuration (which uses two workers per model configuration) is adjusted to use all of the available 32 worker nodes—running 16 parallel configurations instead of 9 at each iteration. Because all configurations are run in parallel in each iteration and the number of iterations is not changed, the total tuning time is roughly the same. The time is slightly longer with 16 parallel configurations because the time for each iteration is determined by the longest running configuration. With many more configurations evaluated (81 compared to 46 by default), more complex models configurations are generated, leading to slightly longer evaluation time, but with the benefit of possibly finding a better model because more candidate configurations were evaluated.

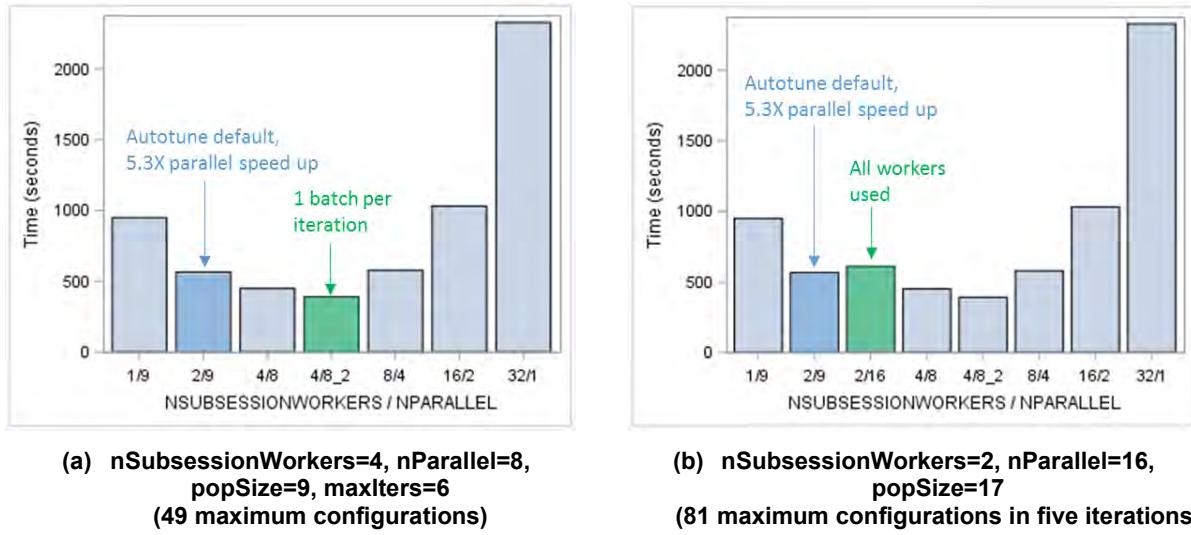


Figure 12. Adjusted Population Size, Bank Data Set

EARLY STOPPING

Some of the tuning actions in the **autotune** action set execute training actions that iterate internally to fit a model, and the maximum number of the internal training iterations is often quite high by default. A high number of training iterations can lead to training times that are longer than necessary and can also lead to overfitting. When model improvement (based on validation error) has stagnated, or has ceased to make more than very minimal improvement in multiple successive iterations as illustrated in Figure 13, it is beneficial to terminate the training at that point. This is referred to as early stopping.

By default, the **tuneGradientBoostTree** action, called by PROC GRADBOOST when the AUTOTUNE statement is included, activates early stopping for more efficient tuning. With gradient boosting, early stopping terminates the training action if no improvement in model error is achieved within the last *n* iterations (*n* is specified in the *stagnation* parameter, which is set to 4 when autotuning). As a result, the actual final number of trees in the reported top model might be less than the best value that the autotuning action selects.

The **tuneNeuralNet** action, called by PROC NNET when the AUTOTUNE statement is included, also activates early stopping for more efficient tuning, but only if the number of internal neural network training iterations is 25 or greater. The *stagnation* parameter here specifies the number of consecutive validations with increasing error rates that are allowed before early termination of the model training optimization process, and the *frequency* parameter specifies how frequently (in epochs) validation occurs during model training. For tuning neural networks, the *stagnation* parameter is set to 4 and the *frequency* parameter is set to 1. An example model training iteration history plot with and without early stopping is shown in Figure 13; clearly most of the improvement is obtained by the early stopping point, which occurs after less than half the number of iterations.

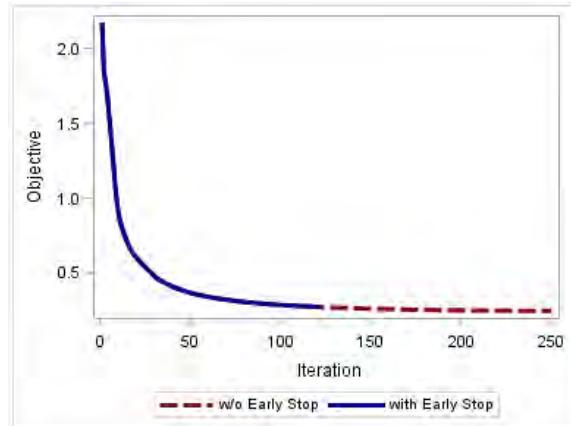


Figure 13. Iteration History Example: Early Stopping versus No Early Stopping

Early stopping can be disabled (allowing all models to train to completion) by specifying a value of False for the *earlyStop* parameter in the **tuneGradientBoostTree** and **tuneNeuralNet** actions or by specifying STAGNATION=0 in PROC GRADBOOST or PROC NNET when the AUTOTUNE statement is included. However, keeping early stopping enabled can often significantly reduce the total tuning time with little effect on the final model accuracy. Final models can be retrained with early stopping disabled to compare accuracy values. Figure 14 shows the reduction of default tuning time and a comparison of final accuracy for tuning a gradient boosting model for a set of benchmark test problems.¹ An average of 40% reduction in tuning time is observed, and the error of the final best model is similar or less with early stopping. Figure 15 shows similar results for tuning a neural network that is trained with 50 iterations of stochastic gradient descent. The average reduction in tuning time is more than 30%. However, in some cases the final model error is higher with early stopping than when the full 50 iterations are run. The model training process can appear stagnated over four epochs, but improvements can occur in later iterations. This is the trade-off and challenge with early stopping. For autotuning, early stopping can first be used to explore more models and refine the search space based on the best models, and then relaxed or disabled to further explore the space around good candidates that were identified. Also, the early stopping parameters can be adjusted, both the STAGNATION value and the FREQUENCY value for neural networks. If the validation checking for stagnation is performed every other epoch (FREQUENCY=2) instead of every epoch, the time savings is reduced to 25%, but the final model error values are closer to those seen without early stopping; these results are shown in Figure 16.

¹ Data sets from http://mldata.org/repository/tags/data/IDA_Benchmark_Repository/, made available under the Public Domain Dedication and License v1.0, whose full text can be found at <http://www.opendatacommons.org/licenses/pddl/1.0/>.

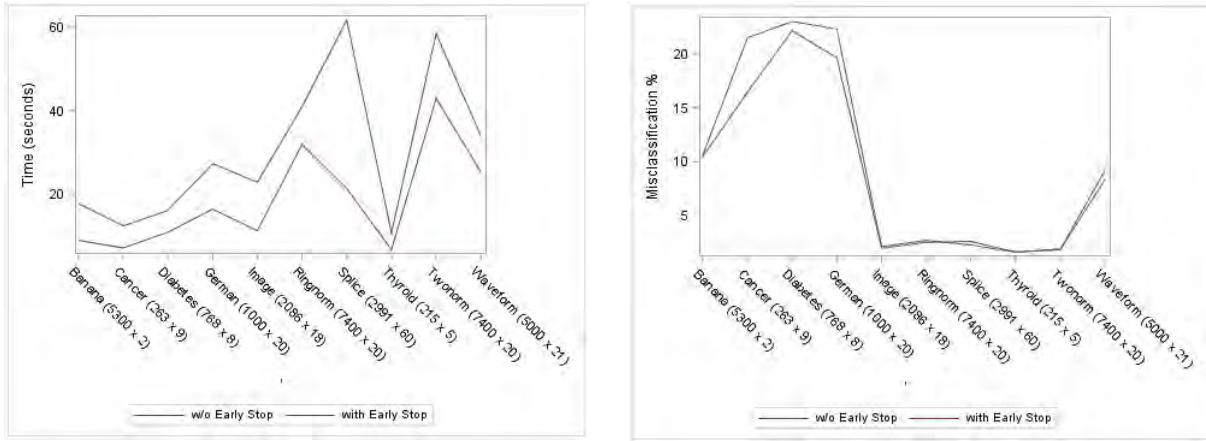


Figure 14. Early Stopping with the tuneGradientBoostTree Action

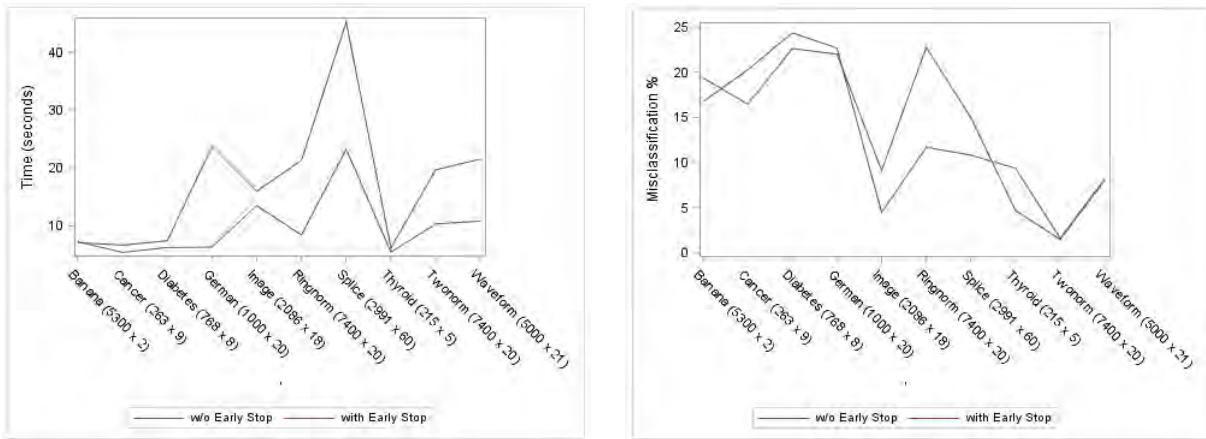


Figure 15. Early Stopping with the tuneNeuralNet Action, Using Stochastic Gradient Descent, 50 Iterations

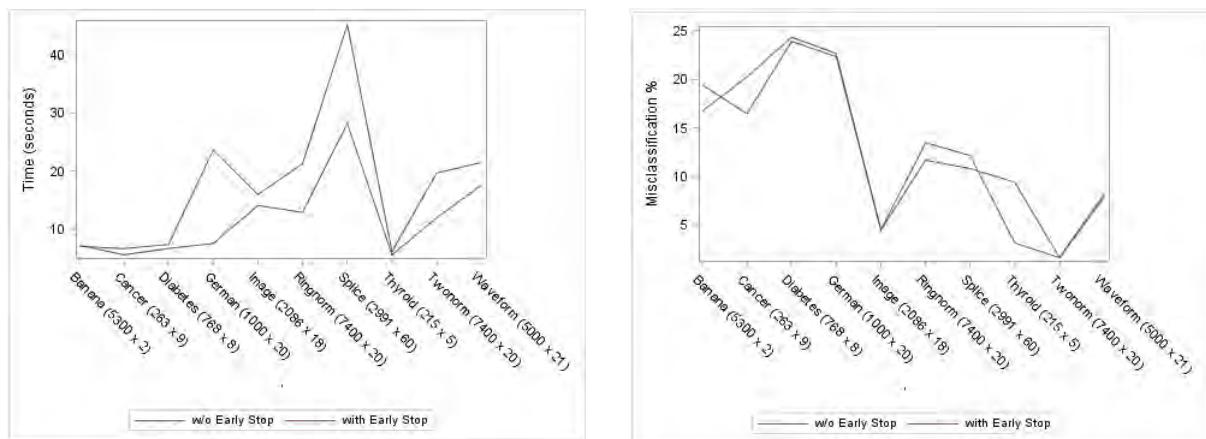


Figure 16. Early Stopping with the tuneNeuralNet Action, Using Stochastic Gradient Descent, 50 Iterations, frequency=2

Case Study Results

Early stopping results for default tuning of gradient boosting models for the four case study data sets are shown in Figure 17, where tuning times with and without early stopping are compared for each data set. Interestingly, you can see different behavior between long narrow data sets and short wide data sets. The early stopping process requires scoring during the model training iterations to determine whether the model error has stagnated. The more observations there are to score, the more the training time increases; models that do not stagnate will take longer to train because of the additional expense of scoring. For models that do stagnate, time can be saved. For the Bank and CoverType data sets, which contain many more observations and fewer columns than the MNIST and CIFAR-10 data sets, early stopping produces no savings. CoverType tuning takes slightly longer with early stopping because of the added cost of validation during tuning. For the MNIST and CIFAR-10 data sets, the validation set is much smaller—10,000 observations compared to more than 318,000 for the Bank data set and more than 174,000 for the CoverType data set. The training cost is also much higher because the MNIST and CIFAR-10 data sets are much wider. As a result, the savings from early stopping outweighs the added cost of validation during tuning. Early stopping for the MNIST data set saves more than an hour of tuning time, a 30% savings. For the CIFAR-10 data, early stopping saves nearly three hours of tuning time, an 18% reduction. Note that this is the default tuning process, which is only 50 maximum configurations.

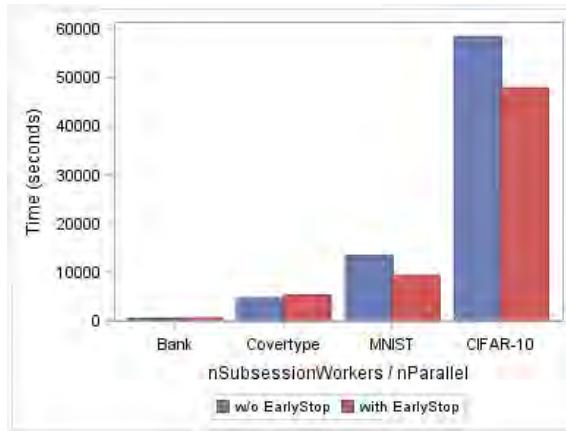


Figure 17. Early Stopping Effect

SUBSAMPLING TRAINING DATA

The expense of model training increases with data set size. In the past, subsampling was commonly used for model training when the data set size was too large. Today, distributed data and distributed training algorithms allow model training to scale to “big data” levels without subsampling. However, as discussed, if more worker nodes are allocated to model training, then fewer nodes are available for parallel tuning of different hyperparameter configurations. To reduce tuning time or increase the number of models that are trained in parallel during tuning within a time budget, autotuning can employ subsampling of the training data. If subsampling of training data is representative of the full data set, a larger number of hyperparameter configurations can be explored more efficiently without diminishing the accuracy of the resulting models. After hyperparameter configuration options have been narrowed, the full training data set can be used for final tuning or to confirm and select among alternate candidate configurations (or both).

The **sampling** action set is used by the autotuning process to create the training and validation partitions if they are not supplied: the **stratified** action is used for nominal type targets (if all target levels can be included in both the training and validation partitions), and the **srs** action is used for a target of interval type and for cases in which stratified sampling is not possible. By default, a validation partition of 30% is used and the remaining 70% is used for model training. Both can be adjusted. For large data tables, tuning efficiency can be increased by subsampling the remaining data for training. For example, 30% of the data can be used for model training and 30% for validation, leaving 40% unused. The training partition size can be specified using the PARTITION statement (specifying both validate and test fractions) in the procedures that include the AUTOTUNE statement, or by using the *trainPartitionFraction* parameter or its alias *trainFraction* with the **autotune** actions. Stratified sampling ensures that the model training and validation partitions are representative of the full data table when possible.

Case Study Results

When the data set is subsampled for more efficient model training, the potential trade-off is reduced accuracy of final best tuned models. This trade-off is illustrated for the four case study data sets in Figure 18. In all cases, the default validation fraction of 30% is used and the training fraction is sampled from 10% to 70%. In all cases except for the CIFAR-10 data set, the axis range for final model accuracy in the plots is 3% so that they can be directly compared; the actual change in accuracy as the training fraction is reduced is less than 3% in these three cases.

In Figure 18(a) the Bank data misclassification error is seen to increase by only 0.4% when the training fraction sample size is decreased from 70% to 10%. However, the tuning time is reduced by more than 35%. When training is performed with 10% of the data, two additional tuning iterations could be added to evaluate 18 more configurations in roughly the same time as when training with 70% of the data, or the population size of each iteration could be increased by three evaluations, from 10 to 13.

The subsampling results for the CoverType data in Figure 18(b) show greater change in the final model error, with an increase of roughly 2.5% when the training fraction is reduced from 70% to 10%. However, note that the default hyperparameter values result in a model with 19% error for this data set; all training sample sizes lead to reduced error with tuning. Also, at a 40% training fraction, a 30% reduction in tuning time is observed with only 0.6% increase in model error. The rate of error increase changes more significantly when 30% or less of the data are used for training.

For the MNIST data set, the final model error increases more rapidly when the training fraction is less than 40%, as shown in Figure 18(c); 2.5% error with 70% training fraction is increased to 4% when only 10% of the data is used for model training. At a 40% training fraction, again only a 0.6% increase in model error is observed. A 14% reduction in tuning time is observed with a 40% training fraction, compared to a 70% training fraction. In this case the time savings is less than it is for the previous two data sets because the data set is very wide, which affects the training time significantly; reducing the number of observations for training has less impact, but savings are still observed.

Both the time change and final model accuracy change are most significant for the CIFAR-10 data set, as shown in Figure 18(d). Here the tuning time is reduced by 23% when using a 10% training fraction compared to 70%, but at a cost of more than 9% in misclassification error. It is known that gradient boosting models are not the best choice for this data set—the misclassification errors are quite high. The increase in error is again slight at first, with the training fraction reduced to 60% or 50%, but increases quickly after that. Also, the time decrease with decreasing fraction is not as smooth as it is for the other data sets. For this complex data set, the default hybrid optimization process, incorporating a genetic algorithm, varies more when the data are changed, especially because solutions are not as good.

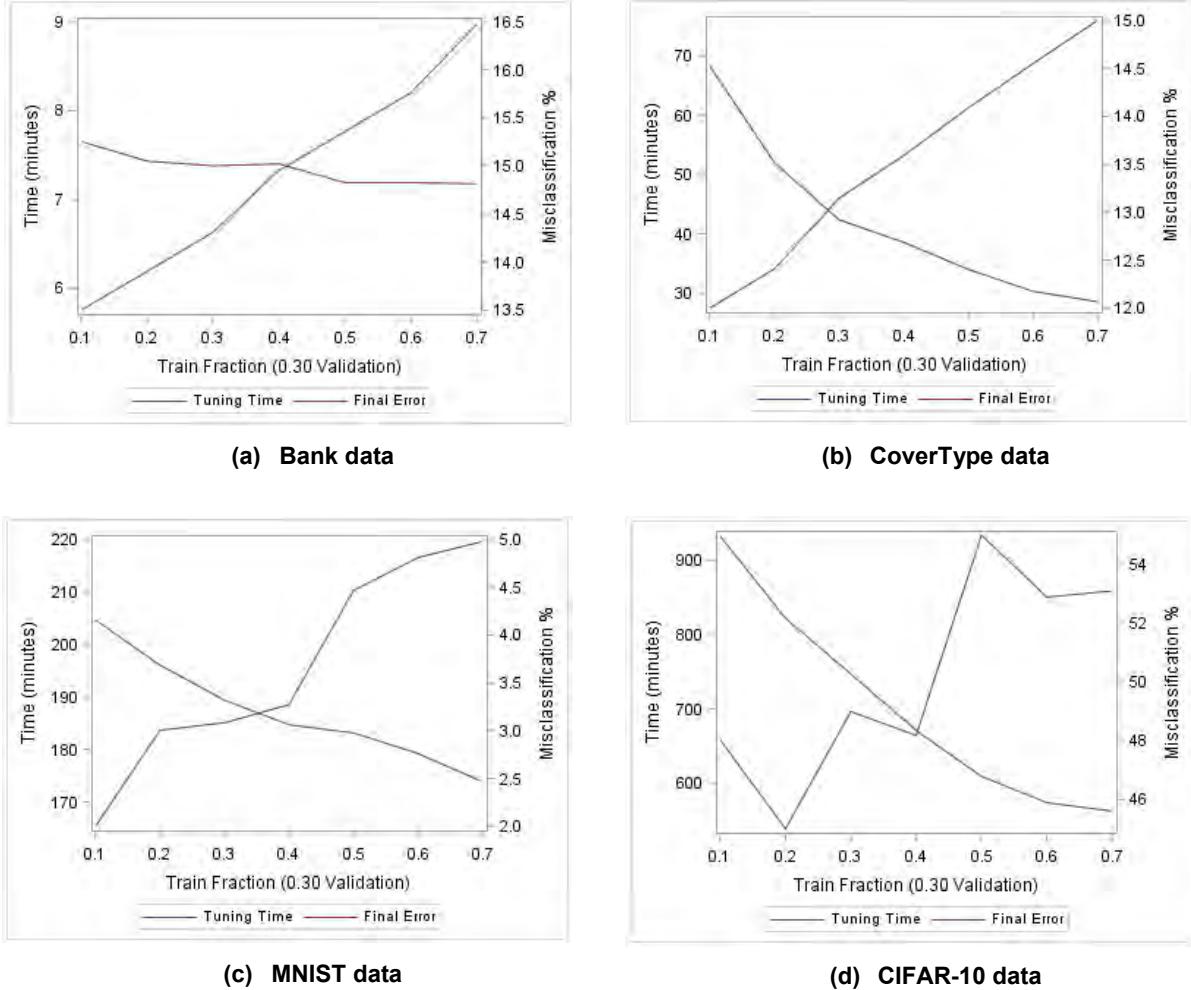


Figure 18. Subsampling Trade-Off

CONCLUSIONS AND RECOMMENDATIONS

It is clear that the optimal number of worker nodes for training a model is not the optimal number when hyperparameters are tuned. Figure 8 through Figure 11 confirm that allocating resources to parallel training of different models reduces the tuning time more than does allocating resources to speed up each model training (that is, using the optimal number of nodes for each model training). Even if a cluster is sufficiently large to support the optimal number of worker nodes for each model configuration and allows all models in an iteration to be evaluated in parallel for the default autotuning process, it might be more effective to increase the number of models that are trained in parallel rather than decreasing the training time, providing a better chance at finding a better model. The options for setting the number of workers for each model training (NSUBSESSIONWORKERS), the number of models trained in each iteration during tuning (POPSIZE), and the number of models trained in parallel (NPARALLEL) can all be used in unison to optimize resource usage and control the hyperparameter tuning expense.

Early stopping can reduce tuning expense, but it can also increase individual training time because it leads to extra validation during the training process. Subsampling can reduce training time, but it can also increase model error; a trade-off is necessary to determine when efficiency is most appropriate (perhaps when exploring a large number of models during tuning) and when accuracy is critical (perhaps after narrowing the choices).

The combination of all these options within the autotuning implementation on SAS Viya can help manage the expense of hyperparameter tuning, allowing more configurations to be evaluated in a specific time budget. Effective settings of these options depend on the specific scenario and the goal of your study. Some best-practice recommendations are offered in Table 3.

Scenario	Suggested Best Practice
RESOURCE ALLOCATION	
If cluster size < population size (GA, Bayesian) or sample size (LHS, random)	<ul style="list-style-type: none"> Set population or sample size as a multiple of $\frac{\# \text{ available worker nodes}}{\# \text{ worker nodes per model (subsession)}}$ (for GA search method, increase population size by 1 to account for the best configuration carryover) This ensures even batches of candidate model configurations, thus no loss of efficiency with a partial batch For GA search method, population size less than the default (10) is not recommended; for example, if cluster size is 8 worker nodes, increase population size to 17 ($8*2+1$) and set maximum iterations or maximum evaluations as desired to limit tuning time
If cluster size > population size (GA, Bayesian) or sample size (LHS, random) and data size is <u>small</u> or tuning time budget is flexible	<ul style="list-style-type: none"> If 1 worker is used for each model (subsession), increase population or sample size to cluster size (increase by 1 for GA search method); this increases the total number of model configurations to be evaluated If multiple workers are assigned to each model (subsession), increase population or sample size to $\frac{\# \text{ available worker nodes}}{\# \text{ worker nodes per model (subsession)}}$ (add 1 for GA search method)
If cluster size > population size (GA, Bayesian) or sample size (LHS, random) and data size is <u>large</u> or tuning time budget is limited (or both)	<ul style="list-style-type: none"> Increase NSUBSESSIONWORKERS to $\frac{\# \text{ available worker nodes}}{\text{population size or sample size}}$ (POPSIZE–1 for GA search method); this will increase the efficiency of training each model during tuning for medium to large data sets Limit NSUBSESSIONWORKERS to 8; in most cases, increasing further will increase training time
EARLY STOPPING	
For initial hyperparameter tuning exploration	<ul style="list-style-type: none"> Early stopping is activated by default when gradient boosting and neural network models are tuned in order to terminate stagnated model training
For refined hyperparameter tuning with narrowed ranges or for confirming final models (or both)	<ul style="list-style-type: none"> Deactivate early stopping or reduce checking frequency (for neural networks, increase the <i>frequency</i> parameter value, which specifies the number of iterations between stagnation checks)

SUBSAMPLING	
If data set is large and is fairly balanced in target	<ul style="list-style-type: none"> Subsampling down to a 40% training fraction is a good trade-off; in most cases, this reduces tuning time by 15–30%, with marginal loss in accuracy Use all data for evaluating final models or for refined tuning
If data set is small	<ul style="list-style-type: none"> Cross validation is preferred over a single validation partition; subsampling of the data is not necessary
If data set is very unbalanced in target or if errors are high	<ul style="list-style-type: none"> Subsampling for the training fraction is not recommended

Table 3. Best Practice Recommendations for Managing Hyperparameter Tuning Expense

APPENDIX A: BENCHMARK DATA SET DESCRIPTIONS

CoverType Data

The CoverType data set is obtained from the UCI Machine Learning Repository (Lichman 2013). The data set, gathered from four wilderness areas in the Roosevelt National Forest, is used to predict forest cover type based on cartographic variables, which include elevation, aspect, slope, horizontal and vertical distance to hydrology, horizontal distance to roadways and fire points, hillshade, specific wilderness area, and soil type. A total of 54 attributes are used to predict seven tree types, making the modeling problem one of multiclass classification. The data set includes 581,012 observations, leading to nearly 32 million values in the complete data set.

MNIST Digits Data

The MNIST (Mixed National Institute of Standards and Technologies) database of handwritten digits (Lecun, Cortes, and Burges 2016) contains digitized representations of handwritten digits 0–9, in the form of a 28×28 image for a total of 784 pixels. Each digit image is an observation (row) in the data set, with a column for each pixel containing a grayscale value for that pixel. After removal of pixels that are blank for all observations, the data set contains 718 attribute columns. The database includes 60,000 observations for training (over 43 million values), and a test set of 10,000 observations. Like many studies that use this data set, this example uses the test set for model validation during tuning.

Bank Data

The Bank data set is a simulated data set that consists of anonymized and transformed observations taken from a large financial services firm's accounts. Accounts in the data represent consumers of home equity lines of credit, automobile loans, and other types of short- to medium-term credit instruments. The data set includes a binary target that represents whether the account contracted at least one new product in the previous campaign season, and 54 attributes that describe the customer's propensity to buy products, RFM (recency, frequency, and monetary value) of previous transactions, and characteristics related to profitability and creditworthiness. With 1,060,038 observations, the data set contains over 57 million values. This data set can be downloaded from GitHub at <https://github.com/sassoftware/sas-viya-machine-learning> (in the data folder).

CIFAR-10 Image Data

The CIFAR-10 data set (Krizhevsky 2009) contains 10 classes of 32×32 color images. The 10 classes are: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. The data set includes 6,000 images per class, with 50,000 images used for training and 10,000 images used as a test set (used for model validation during tuning). The digitized images are represented by a set of RGB (red, green, blue) values for each pixel, resulting in 3,072 attribute columns ($32 \times 32 \times 3$). The data set thus contains over 153 million values.

REFERENCES

- Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). "Algorithms for Hyper-parameter Optimization." In *Proceedings of NIPS*, 2546–2554.
- Bergstra, J., and Bengio, Y. (2012). "Random Search for Hyper-parameter Optimization." *Journal of Machine Learning Research* 13:281–305.
- Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., and Leyton-Brown, K. (2013). "Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters." In *NIPS Workshop on Bayesian Optimization in Theory and Practice (BayesOpt'13)*.
- Koch, P., Wujek, B., Golovidov, O., and Gardner, S. (2017). "Automated Hyperparameter Tuning for Effective Machine Learning." In *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings17/SAS0514-2017.pdf>.
- Krizhevsky, A. (2009). "Learning Multiple Layers of Features from Tiny Images." Technical Report, University of Toronto.
- LeCun, Y., Cortes, C., and Burges, C. J. C. (2016). "The MNIST Database of Handwritten Digits." Accessed April 8, 2016. <http://yann.lecun.com/exdb/mnist/>.
- Lichman, M. (2013). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.
- Wexler, J., Haller, S., and Myneni, R. 2017. "An Overview of SAS Visual Data Mining and Machine Learning on SAS Viya." In *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available at <https://support.sas.com/resources/papers/proceedings17/SAS1492-2017.pdf>.
- Wujek, B., Hall, P., and Güneş, F. (2016). "Best Practices in Machine Learning Applications." In *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. Available at <https://support.sas.com/resources/papers/proceedings16/SAS2360-2016.pdf>.

ACKNOWLEDGMENTS

The Forest Covertype data set is copyrighted 1998 by Jock A. Blackard and Colorado State University. The authors would like to thank Anne Baxter for her contributions to this paper.

RECOMMENDED READING

- *Getting Started with SAS Visual Data Mining and Machine Learning 8.2*
- *SAS Visual Data Mining and Machine Learning 8.2: Procedures*
- *SAS Visual Data Mining and Machine Learning 8.2: Programming Guide*
- *SAS Visual Statistics 8.2: Procedures*
- *SAS Visual Statistics 8.2: Programming Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Patrick Koch SAS Institute Inc. patrick.koch@sas.com	Brett Wujek SAS Institute Inc. brett.wujek@sas.com	Oleg Golovidov SAS Institute Inc. oleg.golovidov@sas.com
--	---	--

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Paper SAS1962-2018

Analyzing Text In-Stream and at the Edge

Simran Bagga and Saratendu Sethi, SAS Institute Inc.

ABSTRACT

As companies increasingly use automation for operational intelligence, they are deploying machines to read, and interpret in real time, unstructured data such as news, emails, network logs, and so on. Real-time streaming analytics maximizes data value and enables organizations to act more quickly. For example, being able to analyze unstructured text in-stream and at the “edge” provides a competitive advantage to financial technology (fintech) companies, who use these analyses to drive algorithmic trading strategies. Companies are also applying streaming analytics to provide optimal customer service at the point of interaction, improve operational efficiencies, and analyze themes of chatter about their offerings. This paper explains how you can augment real-time text analytics (such as sentiment analysis, entity extraction, content categorization, and topic detection) with in-stream analytics to derive real-time answers for innovative applications such as quant solutions at capital markets, fake-news detection at online portals, and others.

INTRODUCTION

Text analytics is appropriate when the volume of unstructured text content can no longer be economically reviewed and analyzed manually. The output of text analytics can be applied to a variety of business use cases: detecting and tracking service or quality issues, quantifying customer feedback, assessing risk, improving operational processes, enhancing predictive models, and many more. SAS® Visual Text Analytics provides a unified and flexible framework that enables you to tackle numerous use cases by building a variety of text analytics models. A pipeline-based approach enables you to easily connect relevant nodes that you can use to generate these models.

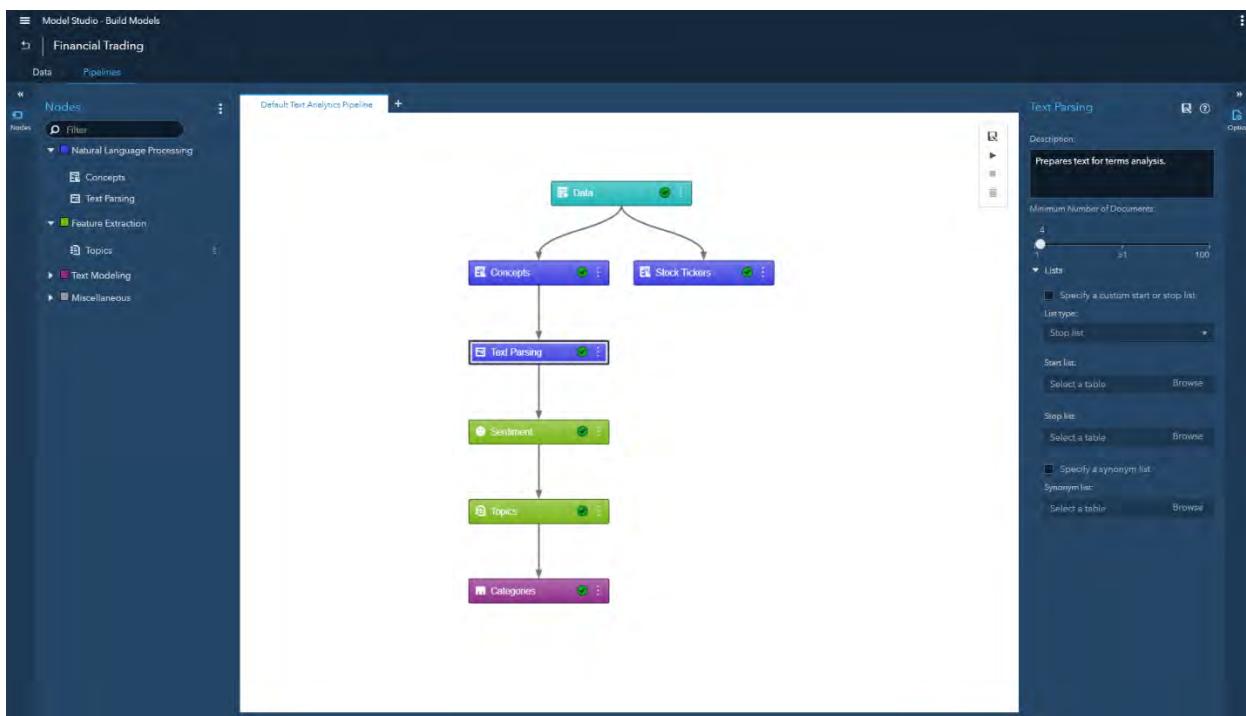
Concepts models enable you to extract entities, concepts, and facts that are relevant to the business. Topic models exploit the power of natural language processing (NLP) and machine learning to discover relevant themes from text. You can use Categories and Sentiment models to tag emotions and reveal insights and issues.

Growing numbers of devices and dependency on Internet of Things (IoT) are causing an increasing need for faster processing, cloud adoption, edge computing, and embedded analytics. The ability to analyze and score unstructured text in real time as events are streaming in is becoming more critical than ever. This paper outlines the use of SAS Visual Text Analytics and SAS® Event Stream Processing to demonstrate a complex event processing scenario. Text models for concept extraction, document categorization, and sentiment analysis are deployed in SAS Event Stream Processing to gain real-time insights and support decision making that is based on intelligence gathered from streaming events.

Big data typically come in dribs and drabs from various sources such as Facebook, Twitter, bank transactions, sensor reading, logs, and so on. The first section of this paper uses SAS Visual Text Analytics to analyze data from trending financial tweets. The latter half focuses on the deployment of text models within SAS Event Stream Processing to assess market impact and intelligently respond to each of the events or data streams as they come in.

EXTRACTING INTELLIGENCE FROM UNSTRUCTURED TEXT USING SAS VISUAL TEXT ANALYTICS

SAS Visual Text Analytics provides a modern, flexible, and end-to-end analytics framework for building a variety of text analytics models that address many use cases. You can exploit the power of natural language processing (NLP), machine learning, and linguistic rules within this single environment. The main focus of NLP is to extract key elements of interest, which can be terms, entities, facts, and so on. Display 1 demonstrates a custom pipeline that you might assemble for a text analytics processing flow. The Concepts node and the Text Parsing node give you the flexibility to enhance the output of NLP and customize the extraction process.



Display 1. Custom Pipeline in SAS Visual Text Analytics

The following list describes the role of each node in this custom pipeline.

- In the Concepts node, you include predefined concepts such as nlpDate, nlpMoney, nlpOrganization, and so on. In this node, you can also create custom concepts and extend the definitions for predefined concepts that are already built into the software. Display 2 shows some custom concepts that have been built to extract information that is related to customer service, corporate reputation, executive appointment, partnerships, and so on, and is likely to affect market trading and volatility. These custom concepts are used for associating categories to each event in SAS Event Stream Processing and will enable automatic concept extraction in future narratives.

The screenshot shows the SAS Model Studio interface for building models. The left sidebar displays a tree view of concepts under 'Financial Trading > Concepts'. Under 'Custom Concepts (12)', there is a node labeled 'CORPORATEREPUTATION'. The main workspace is titled 'Edit a Concept' and contains the following code:

```

1 CLASSIFIER:"hold" rating
2 CLASSIFIER:% stake
3 CLASSIFIER:A01-k match
4 CLASSIFIER:A01-k match
5 CLASSIFIER:A01k match
6 CLASSIFIER:about
7 CLASSIFIER:about business
8 CLASSIFIER:investing
9 CLASSIFIER:stock
10 CLASSIFIER:stocks
11 CLASSIFIER:trading
12 CLASSIFIER:womansentrepreneur
13 CLASSIFIER:worldleading
14 CLASSIFIER:company's initiative
15 CLASSIFIER:about the people who work there
16 CLASSIFIER:access to full time work
17 CLASSIFIER:access to fulltime work
18 CLASSIFIER:access to fulltime work
19 CLASSIFIER:according to market
20 CLASSIFIER:according to the market
21 CLASSIFIER:accountability
22 CLASSIFIER:acquired its partners

```

A green status bar at the bottom indicates 'Code is valid.'

The bottom section shows a preview of the 'body' field from 1484 documents, with several matches highlighted in blue. Examples include:

- ...for #BayArea #smartlock **companies** depends on customers trusting strangers > <https://t.co/m0ZwaBKa11>
- director for Montreal AI lab, **announces plan** to double workforce <https://t.co/oWtcAHPjd> <https://t.co/8t9RanhEnb>
- Stocks** could really ... in 2018: \$AMZN, \$GOOG, \$GOOGL, \$AAPL #StockMarket: <https://t.co/fyQ8imn3u> <https://t.co/y2iXFF2M>
- Stocks rally \$NFLX \$Ostk \$AMZN \$BTC SETH SLTC STSLA #cryptos #Bitcoin <https://t.co/Mx0WE11FW>
- ...on Strength! Among **stocks** that were 🔝 Tuesday, these had the largest net outflows of money. 🔝 <https://t.co/LzStalQJ40>

Document 1 of 1484

Display 2. Concepts Extraction in SAS Visual Text Analytics

In addition, a custom concepts model is also developed to identify stock ticker symbols in each event. This custom concept model is shown in Display 3.

The screenshot shows the SAS Model Studio interface for building models. The left sidebar displays a tree view of concepts under 'Financial Trading > Stock Tickers'. Under 'Custom Concepts (1)', there is a node labeled 'TickerSymbols'. The main workspace is titled 'Edit a Concept' and contains the following code:

```

1 CLASSIFIER:$AMZN
2 CLASSIFIER:$GOOG
3 CLASSIFIER:$GOOGL
4 CLASSIFIER:$AAPL
5 CLASSIFIER:$AAMT
6 CLASSIFIER:$AAPL
7 CLASSIFIER:$AMZN
8 CLASSIFIER:$AXXJ
9 CLASSIFIER:$AXXN
10 CLASSIFIER:$ABAC
11 CLASSIFIER:$ABAK
12 CLASSIFIER:$ABCB
13 CLASSIFIER:$ABCD
14 CLASSIFIER:$ABCE
15 CLASSIFIER:$ABEG
16 CLASSIFIER:$ABENON
17 CLASSIFIER:$ABIL
18 CLASSIFIER:$ABIO
19 CLASSIFIER:$ABRLX
20 CLASSIFIER:$ABMD
21 CLASSIFIER:$ABTK
22 CLASSIFIER:$AMZN

```

A green status bar at the bottom indicates 'Code is valid.'

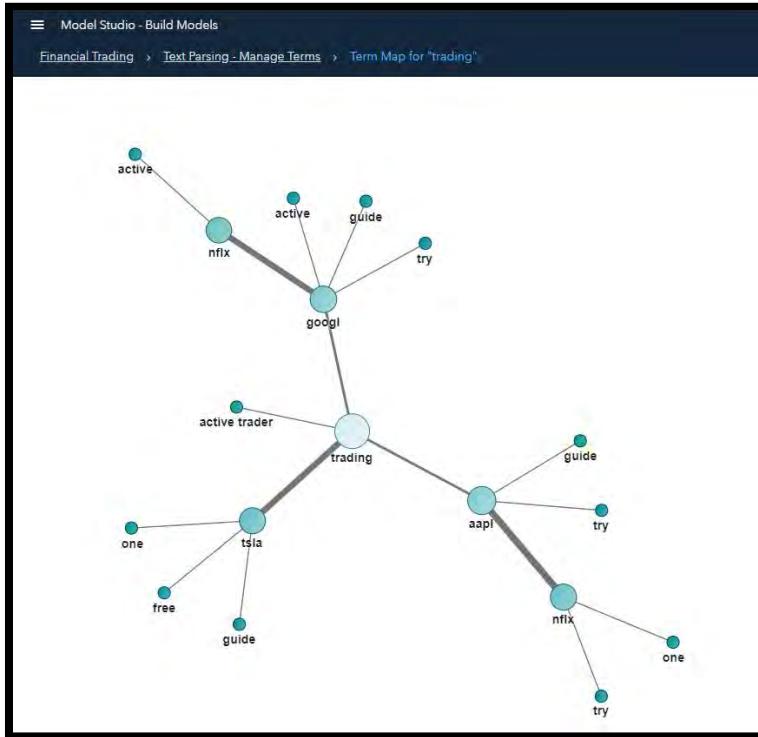
The bottom section shows a preview of the 'body' field from 1566 documents, with several matches highlighted in blue. Examples include:

- ...market know that \$AAPL is coming out with a \$350 smart speaker that can control <https://t.co/2az7AYTCCR>
- ...their acquisition by \$amzn <https://t.co/wte74100Mr>
- Seth \$btc \$tc \$lulu \$nike \$qqq \$t \$smu \$v \$x \$amzn
- \$AMZN went down this road? <https://t.co/h4ALjChawp>
- ...and trade part-time \$AMZN \$NFLX \$GOOGL \$BIIIB \$GS \$FB
- ...stop spending on \$amzn

Document 1 of 1566

Display 3. Extracting Stock Ticker Symbols from Text in SAS Visual Text Analytics

- The Text Parsing node automatically extracts terms and noun groups from text by associating different parts of speech and understanding the context. Recommended lists of Keep and Drop terms are displayed in the interactive window. After the node execution is complete, you can right-click on the node to open the interactive window and drop terms that are not relevant for downstream analysis. The Term Map within the interactive window helps you understand the association of other terms to the term “trading.” See Display 4.



Display 4. Term Map in SAS Visual Text Analytics

- The Sentiment node uses a domain-independent model that is included with SAS Visual Text Analytics. This rules-based analytic model computes sentiment relevancy for each post and classifies the emotion in unstructured text as positive, negative, or neutral. You can deploy the sentiment model in SAS Event Stream Processing to tag emotions that are associated with a post and that might affect trading decisions.
- The final list of terms from text parsing are fed into machine learning for topic detection. In the interactive window of the Text Topics node (see Display 5), you can see commonly occurring themes within a set of tweets. For example, if you select your topic of interest as “+day, options day, 7 day, team, +offering,” the Documents pane shows all the tweets that mention that topic and the terms that exist within that topic, in addition to relevancy and sentiment. You can deploy the Topics model in-stream in order to capture themes as data or events are streaming in. You can also promote topics of interest into your Categories model, which you can deploy in order to classify text into multiple categories. The implementation of this example uses some categories that were created by promoting relevant topics.

The screenshot shows the SAS Visual Text Analytics interface. On the left, the 'Topics' pane displays a list of topics with their document counts, such as 'spy, org, baba, msft, aapl' (1419 documents). On the right, the 'Terms' pane shows 46 of 2857 terms with their relevancy, role, and frequency, including 'day' (0.390, N), 'options day' (0.270, nlpNounGroup), and 'team' (0.270, A). Below these panes is a 'Documents' section titled 'Matched' containing several tweets about stock market news, each with a relevancy score (e.g., 0.998) and sentiment (e.g., red dot for negative). At the bottom, a footer indicates 'Document 1 of 581'.

Display 5. Text Topics in SAS Visual Text Analytics

- In the Categories node, you see the taxonomy (Display 6) that has been designed for document categorization. You can manually extend the auto-generated rules from promoted topics and refer to the previously created concepts within your category rules. You can also use the Textual Elements table to select elements of interest that can be inserted into new rules. Multiple posts or tweets about bankruptcy or layoffs, or about an increase or decrease in the number of shares, often result in stock trading shortly thereafter. This intelligence, if available in real time, can aid in buy or sell decisions that are related to that company.

The screenshot shows the SAS Visual Text Analytics interface. On the left, the 'Categories' pane lists categories like 'All Categories', 'Corporate reputation', 'Customer service', 'Deals', 'Executive appointments', 'Partnerships', 'Firing', 'Bankruptcy or Layoffs', 'Stock fall', and 'Selling or amendment'. The 'Bankruptcy or Layoffs' category is selected. On the right, the 'Edit a Category' pane contains the rule code: '0.01, "bankruptcies", "bankruptcy", "layoffs", "shutdown"'. Below this is a 'Text Sample Test' pane showing several tweets related to company shutdowns and bankruptcies, each with a relevancy score (e.g., 1.000) and sentiment (e.g., red dot for negative). At the bottom, a footer indicates 'Document 1 of 26'.

Display 6. Categorization in SAS Visual Text Analytics

SCORING FINANCIAL POSTS IN REAL TIME TO ASSESS MARKET IMPACT

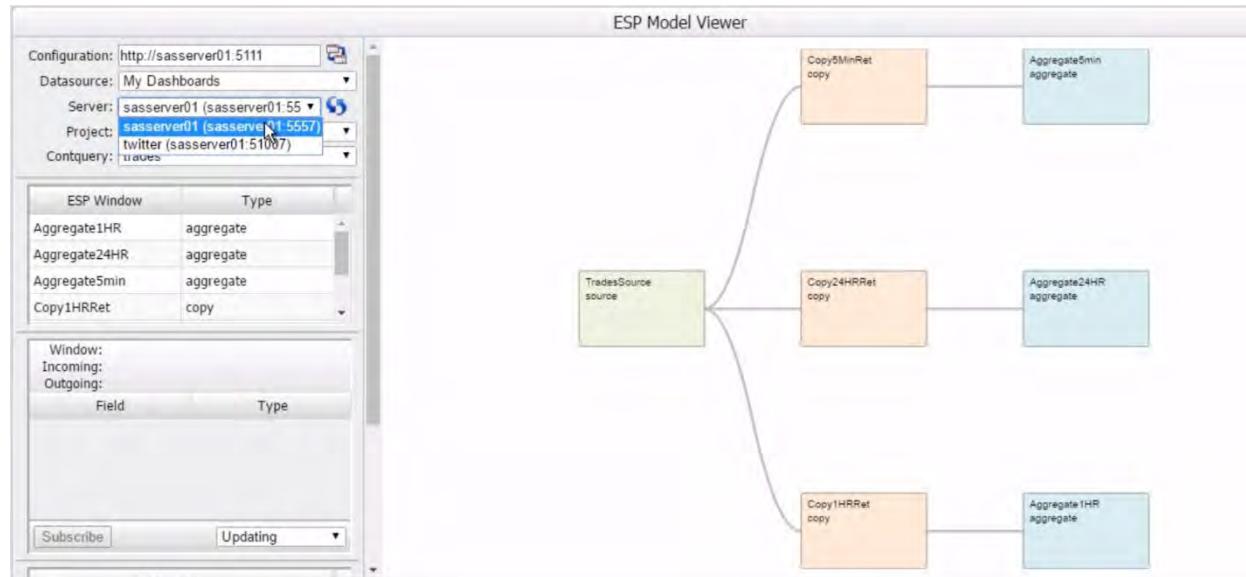
SAS Event Stream Processing is a streaming engine that enables you to analyze or score data as they stream in, rather than first storing them in the database and then analyzing and scoring them in batch. Being able to react to the clicks and events as they are coming in reduces time to action. Event stream processing can occur in three distinct places: at the edge of the network, in the stream, or on data that's at rest (out of the stream).

The SAS Event Stream Processing engine is a transformation engine that augments and adds value to incoming event streams. It is capable of processing millions of events per second. You can perform traditional data management tasks such as filtering out unimportant events, aggregating data, improving data quality, and applying other computations. You can also perform advanced analytic tasks such as pattern detection and text analysis. Events coming in from any source—sensors, Wall Street feeds, router feeds, message buses, server log files—can be read, analyzed, and written back to target applications in real time.

COMPARING STOCK TRADING WEIGHTED AVERAGE PRICE OVER THREE RETENTION PERIODS

The SAS Event Stream Processing studio is a development and testing application for event stream processing (ESP) models. An ESP model is a program or set of instructions that transforms the input event streams into meaningful output event streams. Once the models are built, they can be published into SAS Event Stream Processing for scoring.

In the ESP model presented in Display 7, the Source window (named TradesSource) is reading from one million stock trades, which are all structured data. The three Copy windows define three different levels of event retention: 5 minutes, 1 hour, and 24 hours. The three Aggregate windows create weighted average trade amounts by stock symbol.



Display 7. Model Viewer in SAS Event Stream Processing

The Stream Viewer window in SAS Event Stream Processing provides a dashboard that enables you to visualize streaming events. This example creates three subscriptions for the three aggregate windows, which can be viewed in the dashboard of the Stream Viewer. The dashboard in Display 8 compares the stock trading weighted average price over three retention periods: 5 minute, 1 hour, and 24 hours. The 5-minute view shows what the market is doing right now, whereas the 24-hour view shows what the full day of the market looks like.

Aggregate1HR
sasserver01/tradesDemo/trades/Aggregate1HR

symbol	awap	mi
MJI	100	
RCD	100	
PJA	100	
EUM	100	
PZE	300	
JSM	250	
JPM-X	100	
IMS	100	
HMH	1,600	

Aggregate5min
sasserver01/tradesDemo/trades/Aggregate5min

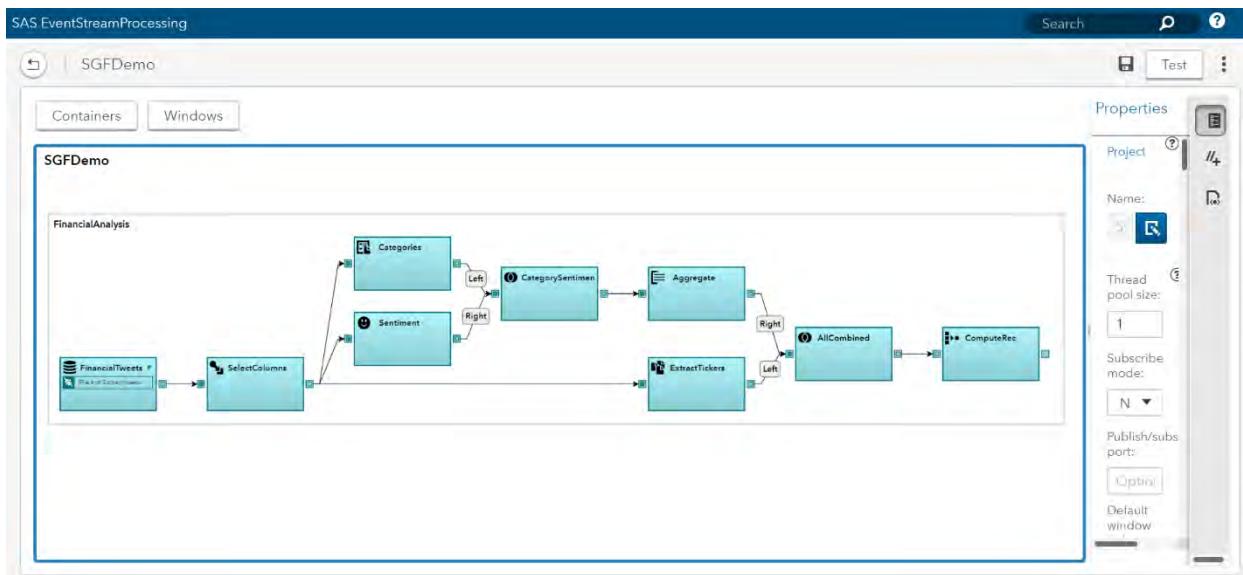
symbol	awap	mi
WXS	35.87	
WLL	93.397	
URTY	98.37	
UNT	39.785	
TPZ	22.125	
TNS	16.06	
TE	16.96	
TAO	18.72	
TAN	7.93	

Display 8. Dashboard Viewer in SAS Event Stream Processing

STOCK RECOMMENDATION BASED ON ANALYSIS OF UNSTRUCTURED TEXT

The models that are built using SAS Visual Text Analytics can applied in batch, in-Hadoop, in-stream, and at the edge. This section uses SAS Event Stream Processing to extract concepts, analyze sentiment about particular companies and their stock, and categorize posts as events stream in real time.

In the process defined in Display 9, tweets are continuously flowing through. The Source window (named FinancialTweets) has a retention policy of 15 minutes, which means that the analysis recommendation is based on the last 15 minutes of captured events. As the tweets come in, they are analyzed: stocks tickers are extracted, sentiment score is assigned, and the content is tagged for appropriate categories.



Display 9. SAS Event Stream Processing Studio

The following list describes each window in Display 9 and its role in the flow.

- Financial Tweets: This is a Source window, which is required for each continuous query. All event streams enter continuous queries by being published (injected) into a Source window. Event streams cannot be published into any other window type. Source windows are typically connected to one or more derived windows. Derived windows can detect patterns in the data, transform the data, aggregate the data, analyze the data, or perform computations based on the data. This example uses a CSV (comma-separated values) file with a small sample of tweets that are related to financial and corporate information. Because the sample is small, the results derived here are purely a proof of concept rather than a true financial analysis for all publicly traded companies. For a true streaming use case, SAS Event Stream Processing provides a Twitter adapter, which can be used to feed tweets in real time.
- SelectColumns: This Compute window enables a one-to-one transformation of input events to output events through computational manipulation of the input event stream fields. You can use the Compute window to project input fields from one event to a new event and to augment the new event with fields that result from a calculation. You can change the set of key fields within the Compute window. This example uses the SelectColumns window to filter out attributes that are not relevant for downstream analysis.
- Categories: This is a Text Category window, which categorizes a text field in incoming events. The text field can generate zero or more categories, with scores. Text Category windows are insert-only. This example uses the model file (.mco) that is generated by the **Download Score Code** option of the Categories node in SAS Visual Text Analytics. Display 10 shows the output that is generated by this window. The output lists the document ID (_Index_ column), category number (catNum column), tagged category (category column), and the relevancy score for assigned categorization (score column).

In [28]: catWindow				
Out[28]:				
		category	score	
Index	catNum			
1	1	Pricing	1.0	
5	1	Pricing	1.0	
10	1	Corporate reputation	1.0	
12	1	Corporate reputation	3.0	
	2	Pricing	1.0	
13	1	Corporate reputation	2.0	
14	1	Corporate reputation	2.0	
15	1	Corporate reputation	1.0	
	2	Pricing	1.0	
18	1	Corporate reputation	1.0	
21	1	Pricing	1.0	
23	1	Corporate reputation	3.0	
	2	Partnerships	1.0	
25	1	Corporate reputation	1.0	
28	1	Partnerships	1.0	

Display 10. Text Category Window Output

- Sentiment: This is a Text Sentiment window, which determines the sentiment of text in the specified incoming text field and the probability of its occurrence. The sentiment value is positive, neutral, or negative. The probability is a value between 0 and 1. Text Sentiment windows are insert-only. This example uses the domain-independent sentiment model file (en-base.sam), which is included in SAS Visual Text Analytics. Display 11 shows the output that is generated by this window. Upon scoring, each document in the _Index_ column is assigned an appropriate sentiment tag (in the sentiment column) along with a relevancy score (in the probability column).

In [24]: sentWindow		
Out[24]:		
	sentiment	probability
Index		
1	Positive	0.600000
2	Positive	0.600000
3	Neutral	0.500000
4	Neutral	0.500000
5	Neutral	0.500000
6	Neutral	0.500000
7	Neutral	0.500000
8	Neutral	0.500000
9	Neutral	0.500000
10	Positive	0.692308
11	Positive	0.600000
12	Neutral	0.500000
13	Positive	0.600000
14	Positive	0.600000
15	Positive	0.600000

Display 11. Text Sentiment Window Output

- CategorySentiment: This is a Join window, which receives events from an input window to the left of the Join window and produces a single output stream of joined events. Joined events are created according to a user-specified join type and user-defined join conditions. This example does an inner join between the category and sentiment tables to create joined events only when one or more matching events occur on the side opposite the input event. Display 12 shows the output that is generated by the CategorySentiment window.

In [25]: catSentWindow				
Out[25]:				
		category	sentiment	probability
Index	catNum			
1	1	Pricing	Positive	0.600000
5	1	Pricing	Neutral	0.500000
10	1	Corporate reputation	Positive	0.692308
12	1	Corporate reputation	Neutral	0.500000
	2	Pricing	Neutral	0.500000
13	1	Corporate reputation	Positive	0.600000
14	1	Corporate reputation	Positive	0.600000
15	2	Pricing	Positive	0.600000
	1	Corporate reputation	Positive	0.600000
18	1	Corporate reputation	Positive	0.600000
21	1	Pricing	Neutral	0.500000
23	2	Partnerships	Neutral	0.500000
	1	Corporate reputation	Neutral	0.500000
25	1	Corporate reputation	Positive	0.600000

Display 12. Joining Category and Sentiment Output Using an Inner Join

- **Aggregate:** Aggregate windows are similar to Compute windows in that non-key fields are computed. Incoming events are placed into aggregate groups such that each event in a group has identical values for the specified key fields. This example aggregates category and sentiment information by stock ticker, as shown in Display 13.

In [26]: resWindow				
Out[26]:				
		term	category	sentiment
Index	termID			
1	1	\$AAPL	Pricing	Positive
5	1	\$AMZN	Pricing	Neutral
10	1	\$NFLX	Corporate reputation	Positive
	2	\$AMZN	Corporate reputation	Positive
12	1	\$AMZN	Corporate reputation Pricing	Neutral
15	3	\$AMZN	Pricing Corporate reputation	Positive
	1	\$AAPL	Pricing Corporate reputation	Positive
	2	\$QQQ	Pricing Corporate reputation	Positive
18	3	\$GOOGL	Corporate reputation	Positive
	2	\$GOOG	Corporate reputation	Positive
	1	\$AMZN	Corporate reputation	Positive
	4	\$AAPL	Corporate reputation	Positive
21	2	\$AMZN	Pricing	Neutral
	1	\$AMZN	Pricing	Neutral

Display 13. Joining Category and Sentiment Output with Extracted Ticker Concepts and Aggregating Categories for Each Stock Ticker

- **ExtractTickers:** This is a Text Context window, which is used here to call the SAS Visual Text Analytics Concepts model to extract key terms or entities of interest from text. Events that are generated from the terms can be analyzed by other window types. For example, a Pattern window could follow a Text Context window to look for tweet patterns of interest. This example combines the extracted tickers with category and sentiment information from posts.

The stock tickers are extracted by using the model file (.li) that is generated by the **Download Score Code** option of the Concepts node in SAS Visual Text Analytics. This file is also shown in Display 3.

- **AllCombined:** This is a second Join window; it combines output from the CategorySentiment window with output from the ExtractTickers window. Display 13 shows the output that is generated by this window. In the AllCombined output, categories and sentiment are aggregated across each stock ticker symbol within a particular document. For example, in document ID 15, \$AMZN refers to both “Pricing” and “Corporate reputation” categories, with the overall sentiment being positive.
- **ComputeRec:** This is a Procedural window, which is a specialized window that enables you to apply external methods to event streams. You can use it when complex procedural logic is required or when external methods or functions already exist. You can apply external methods by using C++, SAS DS2, SAS DATA step, or SAS® Micro Analytic Services. This example calls Python through SAS Micro Analytic Services; the code implements custom logic such as the following:

- If sentiment is “Negative” and relevancy is close to 1, then recommend a sell.
 - If category is “Executive appointments” and sentiment is “Positive,” then recommend a buy.
 - If category is “Corporate reputation” and sentiment is “Positive,” then recommend a hold.
- As events continuously flow into the system, a recommendation is assigned for each event. If the post is associated with negative sentiment, then the recommendation would be to sell the stock. Display 14 shows the output and recommendations that are generated by this window.

In [27]: procWindow		
Out[27]:		
		recommendation
term	_Index_	
\$AAPL	1	HOLD
\$AMZN	5	HOLD
\$NFLX	10	BUY
\$AMZN	10	BUY
	12	HOLD
	15	HOLD
\$AAPL	15	HOLD
\$QQQ	15	HOLD
\$GOOGL	18	BUY
\$GOOG	18	BUY
\$AMZN	18	BUY
\$AAPL	18	BUY

Display 14. Procedural Window Showing Final Recommendation for Each Event

OTHER APPLICATIONS

You can also use SAS Visual Text Analytics and SAS Event Stream Processing to address more mature business use cases, such as the following:

- Financial scenarios
 - Quantitative investment and trading strategies: The trading and investment signals from real-time text analytics are applicable across all trading frequencies and provide an incremental source of quantitative factors.
 - Algorithmic trading: You can enhance algorithmic strategies with automated circuit breakers, or you can develop new algorithms that take advantage of the ability to better predict trading volumes, price volatility, and directional movements.
 - Market making: You can widen spreads or pull quotes when significant negative news is affecting a particular stock.
 - Portfolio management: You can improve asset allocation decisions by benchmarking portfolio sentiment.
 - Fundamental analysis: You can forecast stock, sector, and market outlooks.
- Non-financial scenarios
 - Online email analysis of the mail exchange server to detect intellectual property (IP) leakage as emails are coming inbound and going outbound
 - Fake-news detection and its possible impact on the stock market. Fake news can be identified in various ways: by examining the source, its popularity, and trustworthiness (Waldrop 2017).

CONCLUSION

This paper highlights how unstructured text analysis can be applied in-stream to provide a competitive advantage to financial technology institutions that use the analysis to drive algorithmic trading strategies. Although fintechs use more sophisticated algorithms, this approach demonstrates a simplified implementation that is very feasible within the framework of SAS Visual Text Analytics and SAS Event Stream Processing. This paper does not combine the results of structured data and unstructured text from tweets because access to real-time streaming sources was not available.

In-stream analytics occur as data streams from one device to another, or from multiple sensors to an aggregation point. Event stream processing is also supported at the “edge” so that you can analyze any data that are processed on the same device from which they are streaming.

REFERENCE

- Waldrop, M. Mitchell. 2017. “News Feature: The Genuine Problem of Fake News.” *Proceedings of the National Academy of Sciences of the United States of America* 114 (48): 12631–12634.
<http://www.pnas.org/content/114/48/12631>

ACKNOWLEDGMENTS

The authors thank Kevin Smith for helping implement the workflow using a Python interface to SAS Event Stream Processing.

They also thank Anne Baxter for editing the paper.

RECOMMENDED READING

- *SAS® Visual Text Analytics: Programming Guide*
- *SAS® Event Stream Processing: Programming Reference*
- Robert, Nicholas. “How to perform real time text analytics on Twitter streaming data in SAS ESP.” Available <https://blogs.sas.com/content/sgf/2016/10/05/how-to-perform-real-time-text-analytics-on-twitter-streaming-data-in-sas-esp/>. Last modified October 5, 2016. Accessed on February 26, 2018.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Simran Bagga
SAS Institute Inc.
simran.bagga@sas.com

Saratendu Sethi
SAS Institute Inc.
saratendu.sethi@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Harvesting Unstructured Data to Reduce Anti-Money Laundering (AML) Compliance Risk

Austin Cook and Beth Herron, SAS Institute Inc.

ABSTRACT

As an anti-money laundering (AML) analyst, you face a never-ending job of staying one step ahead of nefarious actors (for example, terrorist organizations, drug cartels, and other money launderers). The financial services industry has called into question whether traditional methods of combating money laundering and terrorism financing are effective and sustainable. Heightened regulatory expectations, emphasis on 100% coverage, identification of emerging risks, and rising staffing costs are driving institutions to modernize their systems. One area gaining traction in the industry is to leverage the vast amounts of unstructured data to gain deeper insights. From suspicious activity reports (SARs) to case notes and wire messages, most financial institutions have yet to apply analytics to this data to uncover new patterns and trends that might not surface themselves in traditional structured data. This paper explores the potential use cases for text analytics in AML and provides examples of entity and fact extraction and document categorization of unstructured data using SAS® Visual Text Analytics.

INTRODUCTION

Financial Institutions dedicate substantial resources in support of government's efforts to curb money laundering and terrorism financing. Money laundering is the process of making funds that were gained through illegal channels appear legitimate, typically through a process of placement, layering, and integration. Terrorism financing is often more challenging to identify, as the funding can be raised through legitimate means, but later used to fund an act of terror or support a terrorist organization. Detecting these patterns can often feel like a game of "whack-a-mole;" by the time a new control is implemented to identify a known risk, the criminals have already changed their behavior to elude your efforts. The stakes are high, as the amount of money laundered per year is estimated to be 2 to 5% of global GDP. That's 2 trillion in USD according to the United Nations Office on Drugs and Crime ([UNODC](#)). In today's big-data environment, using modern technology to quickly identify financial crimes is critical.

A lot has changed globally since the early AML regimes of the 1970s. A growing regulatory landscape has led to higher penalties for program deficiencies. Banking has fundamentally changed with the creation of digital channels, faster payments, and new financial instruments. Data storage has become cheaper, opening the opportunity to process big data rapidly. Financial institutions have mostly adapted to these changes through enhancements to their rule-based detection programs and, as a result, have seen their headcount and costs soar. There's an appetite to overhaul the system to reduce false positive rates, increase the detection of money laundering, and automate many of the tedious tasks required in the investigations process. With the help of SAS® Visual Text Analytics, we can leverage artificial intelligence techniques to scale the human act of reading, organizing, and quantifying free-form text in meaningful ways, uncovering a rich source of underused risk data.

UNSTRUCTURED DATA SOURCES

While structured data such as transaction, account, and demographic information has been used in combating money laundering for years, financial institutions are just now beginning to see the value in harvesting unstructured data sources. These data sources are both vast and rich with valuable information that provides new data points, creates linkages, and identifies trends. Here is a list of the more notable sources of unstructured data that can be used for AML:

- **Wire Data** - Wire transfers between financial institutions contain much more valuable information than just the amount of money being sent. Along with origination, intermediary, and beneficiary data, wires

often include free-form text including payment instructions and other messaging.

- **Transaction Review Memos** - The branch employees and client managers are the first line of defense when it comes to protecting the bank from money laundering. Typically, these individuals report valuable insight to the AML group through a transaction review memo. The details included in these memos are at the branch attendee's discretion, but often they have supporting detail on why the transaction was deemed suspicious that might not be apparent in the transaction alone.
- **Case Data** - Anti-money laundering case data contains information enriched by the investigator during the life of the investigation. Cases generally contain several free-form text fields including notes, comments, and email correspondence as well as a narrative report explaining the final disposition. If suspicious activity is identified, a suspicious activity report (SAR) will be filed.
- **Suspicious Activity Report Data** - SARs are documents that financial institutions must file with their in-country government agency following the identification of potential unusual behavior related to money laundering or fraud. These documents are typically free-form text and generally contain several pieces of information about the person, company, and entity or entities of interest; the general findings from the investigator as to what the suspicious activity was; as well as any supporting evidence for the suspicious activity.
- **Negative News** - Beyond unstructured data your financial institution generates, there is a vast amount of publicly generated data from news and media organizations. Public news data can be used to identify supporting information about your customers including relationships to businesses or risky behaviors and criminal activity.
- **Email/Phone/Chat** - In addition to transactional data, risk factors might be identified in the non-transactional data stored by the bank in the form of email, phone, or chat conversations between the customer and employee.
- **Law Enforcement Requests** - Financial institutions have an obligation to identify subjects of law enforcement requests and file SARs where appropriate. Grand jury subpoenas, national security letters, and other requests are received in electronic format and contain text regarding persons of interest and requests for information.
- **Trade Documents** - The global trade system remains primarily a paper-based system. The trade documents (letters of credit, bills of lading, commercial invoices, other shipping documents) contain critical risk information in free-form text such as boycott language, dual use goods, inconsistent unit pricing, and other trade-based, money-laundering vulnerabilities.

USE CASES IN AML

Mining your unstructured data can be valuable in uncovering new insights to help combat money laundering in your financial institutions. Processing techniques such as theme detection, categorization, and entity or fact extraction are all ways to provide structure to free-form text. Once text is structured, there are several use cases to apply this data to ensure compliance:

- **Negative News Monitoring** - As an industry standard, financial institutions typically look for negative news related to high-risk customers and customers who have an open AML case. With the wide array of digital news made available daily, the identification of credible news can be challenging. Negative news not relevant to compliance can bias an investigator's decision process, while missed news can leave an institution open to reputational risk. Coupled with bank policy and risk tolerance, an automated process to identify negative news and successfully link this information to customers provides both cost and time savings through automation.
- **Network Analytics** - Perhaps one of the best pieces of information for investigating AML is to understand relationships among your customers, as well as non-customers. Most institutions have structured data for known relationships among their customers, but often there are gaps with unknown relationships and those relationships with non-customers. Relationships and networks often surface through normal investigative procedures and are documented in case notes and SAR data. Storing this valuable information and displaying it for future use along with geographic tagging

provides deeper insights to the investigations process.

- **SAR Attribution Detection** - The detection of money laundering is an exercise in correctly identifying rare events in vast amounts of data. As the AML compliance industry starts to explore the application of artificial intelligence and machine learning to replace Boolean rules, the need for reliably labeled data (target variables) for training becomes even more important. Often, SARs are filed based on external information, but are attributed to the success of one or more rule-based scenarios. Text mining can help determine the correlation. This is critical to not only tune existing models, but also to allow banks to predict newly identified patterns in the future.
- **Trade Finance Document Categorization** - Deciphering trade documents is a tedious, manual process. We've been testing cognitive computing capabilities that are used for character recognition and natural language processing for document categorization. In a pilot with a tier 1 bank, our models read trade finance documents with ~99% accuracy and reduced the time to manually process the documents from several weeks to 26 seconds in an automated process.

EXAMPLE FRAMEWORK USING SAS® VISUAL TEXT ANALYTICS

This paper explores the process of processing unstructured data to support any of the use cases listed above. To demonstrate the potential applications, we will follow the framework below, primarily using SAS Visual Text Analytics as the enabling technology.

- **Data Acquisition** – Data is acquired for the example use case utilizing web scraping tools and is imported into SAS Visual Text Analytics.
- **Concept Extraction** – Predefined and customized concepts are generated to extract key facts from the unstructured data.
- **Text Parsing** – The individual records are parsed to enumerate the terms contained in the documents and apply filtering with start and stop lists.
- **Topic Generation** – Individual records are grouped into a collection of related themes containing similar subject matter automatically based on a bottom-up approach using the underlying terms.
- **Categorization** – Documents are classified into predetermined categories based on a top-down approach of the areas of interest using linguistic rules.
- **Post-Processing** – Output from SAS Visual Text Analytics is processed and prepared for use in modeling or investigative tools.

DATA ACQUISITION

While SAR information is not publicly available, we wanted to conduct our analysis on text data with similar content and format. The Internal Revenue Service (IRS) publishes summaries of significant money laundering cases each fiscal year, dating back to 2015. This data is rich with information, including people, organizations, risk typologies, locations, and other interesting data related to financial crimes. Below is an example of an IRS case from our data set:

“Former Owners of Money Transmitter Business Sentenced for Conspiring to Structure Financial Transactions

On October 25, 2016, in Scranton, Pennsylvania, German Ossa-Rocha was sentenced to 27 months in prison and two years of supervised release. On October 26, 2016, Mirela Desouza was sentenced to 18 months in prison and two years of supervised release. Ossa-Rocha and Desouza were the former owners of Tropical Express, a money transmitter service business located in Stroudsburg. Beginning in approximately January of 2008 and continuing through December 2011, Ossa-Rocha and Desouza structured financial transactions that represented the proceeds of drug trafficking in a manner intended to avoid reporting and recording requirements. The amount of funds involved in the structuring was approximately \$340,000. The funds were transmitted by Ossa-Rocha and Desouza via wire transfers to the Dominican Republic.” ([IRS](#))

Web scraping tools were used to extract the various money laundering examples and write to a CSV file with four columns: observation number, year, title, and text narrative. The CSV file was then imported into SAS Visual Text Analytics for analysis.

CONCEPT EXTRACTION

After initializing a project and loading the data, the first step in the process was focused on concept and fact extraction. With our data being rich in entities and facts, we wanted to extract these from the text for potential use in further analysis and research by investigators. In our model pipeline, this was done by dragging a Concept node and placing it on top of the Data node. SAS Visual Text Analytics comes with predefined concepts out of the box, as well as the ability to write your own custom concepts using LITI (language interpretation and text interpretation) syntax. For our analysis, we enabled the predefined concepts and wrote several custom concepts that are highlighted below.

The predefined concepts are common points of interest in which the rules come out of the box to immediately apply to your data, saving you time and helping you gain instant insights. Here are the predefined concepts of interest for our analysis:

- **nlpDate** – Identifies and extracts all dates and date ranges in your data in several formats (for example, May 2003, 05/15/2007, between 2007 and 2009, and so on).
- **nlpMeasure** – Identifies and extracts measures of time and quantities (for example, 30 years, 500 kilograms, and so on).
- **nlpMoney** – Identifies and extracts all references to currencies (for example, \$272,000, more than \$3 million, and so on).
- **nlpOrganizations** – Identifies and extracts all organization names (for example, U.S. Treasury, Department of Agriculture, and so on).
- **nlpPerson** – Identifies and extracts all names (for example, Joyce Allen, Robert L. Keys, and so on).
- **nlpPlace** – Identifies and extracts all places (for example, Asheville, North Carolina, Newport Beach, California, and so on).

Error! Reference source not found. below shows a set of matched concepts for the predefined concept nlpMoney.

The screenshot shows the SAS Visual Text Analytics interface with the following details:

- Documents:** 180
- Test Sample Text**
- Filter:** All → Matched
- Category:** narrative
- Text Preview:**
 - ...pay restitution of **\$25,434,939** to his victims. In August 2015, Hernandez pleaded guilty to money laundering. According to court documents, Hernandez managed an estimated **\$80 million** investment portfolio of a Mexican businessman and his family from 2005 to 2014, but had not invested his clients' money like he had agreed. Hernandez spent their money to purchase homes, planes and other assets for his personal use.
 - ...ordered to pay **\$2,502,348** in restitution to the IRS. Collado previously pleaded guilty to theft of government funds. According to court documents, Collado operated two grocery stores. In 2012, co-conspirators brought Rodriguez fraudulently obtained U.S. Treasury checks, which Rodriguez cashed without receiving identification for the individuals listed on the checks, or any other source of authority for the co-conspirators to cash the checks. Rodriguez initially was not a licensed check casher, but one co-...
 - ...his participation in a **\$300 million dollar** vacation rental fraud scheme. In addition, forfeiture money judgments were entered against Clark that includes **\$303,800,000** for the bank fraud and **\$3,300,000** for the SEC obstruction. There is also court-ordered forfeiture of specific overseas assets of approximately **\$2.6 million**. Clark was convicted on Dec. 11, 2015, of three counts of bank fraud and three counts of making a false statement to a financial institution. According to court records, the scheme involved sales at Cay...

Figure 1. Matched Concepts for Predefined Concept nlpMoney

While the predefined concepts are valuable in and of themselves, they are also useful for referencing in your custom concepts. An example of this can be seen with our custom concept Fine_Amount. The predefined concept nlpMoney will extract out all references to money, but suppose we want to exclusively extract out the fines associated with each record for further analysis. Instead of filtering through all references to money, we can define a custom concept to pull out only currencies associated with a fine.

Figure 2 below shows the LITI syntax to generate this rule:

```
1 C_CONCEPT:ordered to pay _c{nlpMoney}
2 C_CONCEPT:ordered to forfeit _c{nlpMoney}
3
4
```

Code is valid.

Figure 2. Custom Concept Fine_Amount LITI Syntax

The Fine_Amount custom concept uses the C_CONCEPT rule, which enables you to return matches that occur only in the context that we desire. In our case, we want to return the currency found by the nlpMoney predefined concept, but only in the context of a fine as in “ordered to pay” or “ordered to forfeit”.

A set of custom concepts was built on top of the predefined concepts to extract additional useful facts that could be helpful for indexing and searching, as well as additional analysis. Table 1 below summarizes the custom concepts that were developed, the type of concept used, and an example of the output.

Custom Concept	Concept Type	Example Output
Drug_Names	CLASSIFIER	Marijuana
Prison_Sentence	C_CONCEPT	60 months
Drug_Amount	CONCEPT_RULE	15 kilograms
Investment_Fraud_Amount	CONCEPT_RULE	\$200 million
Investment_Fraud_Victims	CONCEPT_RULE	70 victims
Case_Charges	CLASSIFIER	Identity theft
Sentence_Location	CONCEPT_RULE	Providence, Rhode Island

Table 1. Custom Concept Definitions

TEXT PARSING

The next step in our analysis was to parse the text and create our term document matrix. In our model studio pipeline, this is done by dragging the Text Parsing node and placing it on top of the Concept node. SAS Visual Text Analytics allows you to customize how terms are parsed by configuring the minimum number of documents the term must be found in to be included for analysis, as well as using custom start, stop, and synonym lists. For the purposes of our example, we used the Text Parsing node to further explore some terms of interest for additional context and understanding. Figure 3 is an example of a term map used for exploration purposes.

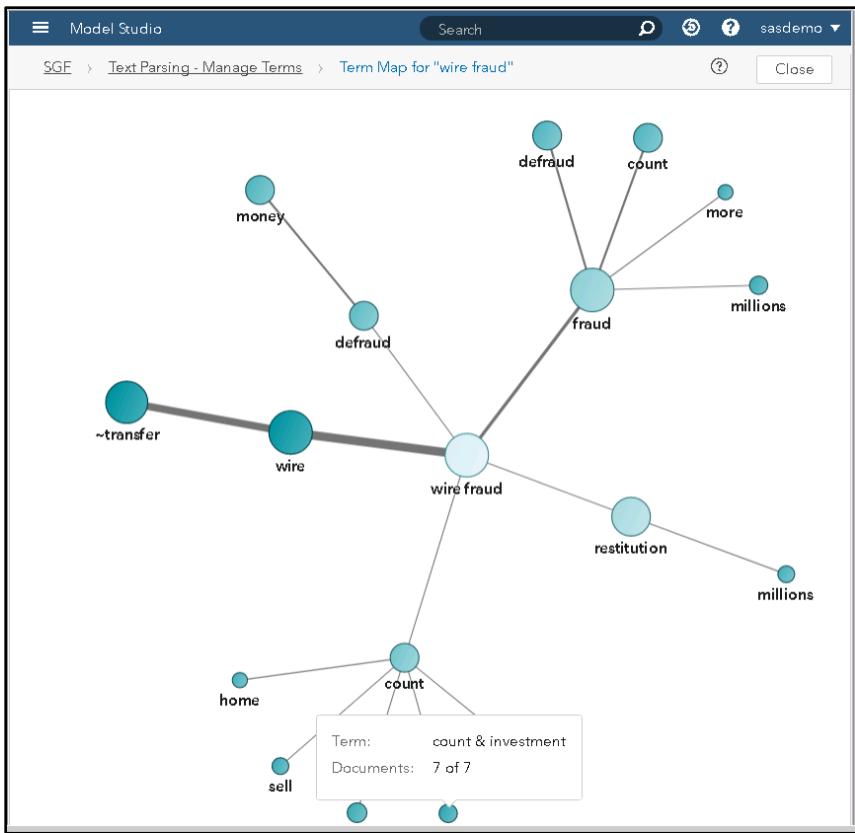


Figure 3. Term Map for “wire fraud”

TEXT TOPICS

Continuing with our analysis, we wanted to understand any relevant themes found in the data with the underlying terms that were parsed. For this, we dragged a Topic node and placed it on top of the Text Parsing node. SAS Visual Text Analytics allows you to automatically generate topics or choose the number of topics to generate, as well as set several other configurations including the term and document density. With a few iterations, we found the most informative results by setting the number of topics generated at 20, as well as term and document density of 2 and 1, respectively. Here is the output of the text topics.

	Topic	Documents
□	+investor, +investment, +invest, capital, +return	34
□	cocaine, cocaine, +possess, +residence, +kilogram	28
□	marijuana, california, +sale, +drug, marijuana	28
□	+victim, costa, costa rica, rica, +co-conspirator	28
□	+church, +client, plan, boston, +asset	26
□	lee, +victim, portland, +live, oregon	25
□	+loan, +false statement, +statement, bank fraud, false	24
□	+check, +cash, +refund, +tax, +check	23
□	+report, +avoid, +structure, +casino, cash	23
□	+request, information, +order, +purchase, +supply	22
□	+stock, shell, u.s., arrest, +trade	21
□	equipment, +steal, carolina, north carolina, unlawful	21
□	fictitious, +employee, +client, +company, +create	20
□	+prescription, +patient, oxycodone, +physician, +substance	17
□	jr., +dollar, diego, united, san	17
□	+buyer, +mortgage, straw, +straw buyer, +application	16
□	construction, +bond, +bond, +contract, +project	16
□	law firm, firm, +law, +client, marijuana	16
□	silk road, silk, road, +user, +website	12
□	reserve, liberty, liberty, reserve, +user	9

Figure 4. Text Topics and Associated Document Count

Upon inspecting the topics, we were interested in two themes that were promoted to categories for ongoing analysis. The topics that were automatically generated provided a new lens on the data that we would like to track further and categorize new documents moving forward.

Topic Terms	Topic Theme	Percent of Documents
+buyer, +mortgage, straw, +straw buyer, +application	Real Estate Investment Fraud	9.4%
silk road, silk, road, +user, +website	Dark Web Drug Trade	7.0%

Table 2. Text Topics Promoted to Categories

TEXT CATEGORIES

Previously, we discussed text topics and the bottom-up approach of using the underlying terms to generate topics of interest. Our next step in our analysis was to take a top-down approach and define categories of interest using linguistic rules available in SAS Visual Text Analytics. In our model pipeline, this is done by dragging a Category node and placing it on top of the Topic node.

Categorizing your documents can be valuable for several reasons, such as creating tags for searching or for assigning similar documents for workflow purposes. Previously, we identified two categories of interest that we converted from the topics that were generated using the Topic node. In addition to these, we created a custom hierarchy of categorization that will help with future analysis. The table below shows the hierarchy of categories we were interested in.

Level 1	Level 2	Percentage of Matches
Drug Activity	Pharma Drugs	3%

	Illegal Drugs	15%
High Risk Customer Groups	Casino	3%
	Real Estate	23%
	Shell Company	3%
Financial Crime Charges	Bank Fraud	14%
	Bulk Cash Smuggling	4%
	Check Fraud	1%
	Identity Theft	6%
	Investment Fraud	8%
	Mail Fraud	16%
	Structuring	3%
	Tax Fraud	5%
	Wire Fraud	28%

Table 3. Custom Category Matches

Each category uses Boolean and proximity operators, arguments, and modifiers to effectively provide matches to only desired documents. Through the authors' domain expertise and the capabilities of SAS Visual Text Analytics, we were able to provide relevant matches on several categories of interest. An example of this concept is outlined below using the text category for the custom category "Identify Theft":

The screenshot shows the SAS Visual Text Analytics interface. At the top, there is a search bar with the query: `(OR, "identity theft", (SENT, (OR, "identity@", (DIST_3, "personal@", "information@")), (OR, "split@", "dual", "stole@", "fabricate@", "obtain@"))))`. Below the search bar, a message says "Code is valid." A "Documents" section shows a list of matched results under the "Matched" tab. The results are listed in a table with columns for "narrative" (the document snippet) and "Relevancy" (the score). There are five matched documents:

- ...to harbor aliens, **identity theft**, conspiracy to commit health care fraud and filing false claims. According to court documents, Khdeer was the last of 18 people to be sentenced for their roles in a series of criminal schemes that centered around seven IHOP restaurants owned by Tarek "Terry" Elkafrawi in northwest Ohio and Indiana. The schemes resulted in losses of more than \$3 million. In 2008, the Findlay IHOP burned as the result of arson started by a co-conspirator at the direction of Elkafrawi and Khdeer to facilitate an... Relevancy: 6.000
- ...wire fraud, aggravated **identity theft** and money laundering. According to court documents, from about September 2011 to January 2014, Sanders and others conspired to defraud state unemployment offices in Ohio, California, North Carolina, Massachusetts and Illinois. Sanders fraudulently obtained personal identifying information from unsuspecting individuals to submit fraudulent claims for unemployment insurance benefits. Sanders also created state unemployment insurance accounts for multiple fictitious... Relevancy: 6.000
- ...funds and aggravated **identity theft** in connection with his participation in a scheme to cash more than \$400,000 in fraudulently obtained federal tax refund checks issued in other people's names. According to court documents, Mejia worked at branches of a bank in Yonkers and Manhattan. Mejia initially was a banker and later became the branch manager of multiple branches of the bank. From 2010 through 2013, Mejia participated in a scheme to fraudulently obtain and cash tax refund checks issued by the United... Relevancy: 4.000
- ...returns filed using **stolen identities**. Wheeler made false entries on the face of the checks to make it appear as if she received identification when the checks were cashed, when in fact, she never received any forms of identification. In total, Wheeler received and cashed approximately 361 checks totaling \$780,760 in tax refunds. Relevancy: 3.000
- ...checks had been obtained through fraud and misused other people's identities. Linares also knowingly failed to follow the requirements placed on him as a registered money service business and failed to prevent his store from being used to facilitate criminal activity and launder money. Finally, Linares attempted to obstruct efforts by the IRS in efforts to determine whether he was following the law. Relevancy: 2.000

Figure 5. Text Category for "Identity Theft" with Matched Output

The "Identity Theft" rule can be broken up into two main components using the OR operator. The first component is simply looking for a direct match for the two sequential terms "identity theft", which provides several simple matches in the output found in the bottom of Figure 5. The second component uses the

SENT operator and will trigger a match if two sub-components exist in the same sentence somewhere within the document. The first sub-component is looking for some form of the word “identity” or a close combination of “personal” and “information”. The second sub-component is looking for the action of theft including terms such as “split”, “dual”, “stole”, “fabricate”, or “obtain”. The fourth and fifth matches in Figure 5 highlight the types of matches this will create in the form of “stolen identities” and “obtained identities” in the fourth and fifth match, respectively.

POST-PROCESSING

Once your project is set up in SAS Visual Text Analytics, you can produce score code and apply this to new data for ongoing tracking and monitoring. There are several types of post-processing that can happen depending on your use case and what the type of output you are working with. The most common types of post-processing can be found below:

- **Categorical Flags** – Typically, the presence or match for a category is used as a binary indicator for each document and can be used in filtering or searching, or as inputs to machine learning algorithms.
- **Network Analysis** – Extracted concepts such as locations, people, and organizations can be post-processed to show linkages and used as input to network diagrams for analysis.
- **Numerical Analysis** – Extracted concepts such as duration, fine amounts, or other numerical fields extracted from the documents can be post-processed to derive summarizations and averages of areas of interest.

CONCLUSION

There is a lot of excitement in the financial crime and compliance industry around the application of artificial intelligence and automation techniques. We see many opportunities available today to apply these methods to improve the effectiveness of detection programs and automate the manual tasks being performed by investigators. Text analytics is one area that has enormous potential, given that compliance departments have vast amounts of untapped, unstructured data sources. These sources contain rich information including who, where, what, when, and how that can be used as an input to many financial crimes use cases such as Negative News Monitoring, Trade Finance Monitoring, and SAR/STR Quality Assurance. With SAS Visual Text Analytics, banks can extract and derive meaning from text and organize it in a way that helps them perform these complex tasks that were previously accessible only through manual human review.

REFERENCES

UNODC (United Nations Office on Drugs and Crime). n.d. “Money-Laundering and Globalization.” Accessed February 20, 2018. Available <https://www.unodc.org/unodc/en/money-laundering/globalization.html>.

IRS (Internal Revenue Service). 2017. “Examples of Money Laundering Investigations - Fiscal Year 2017.” Accessed February 20, 2018. Available <https://www.irs.gov/compliance/criminal-investigation/examples-of-money-laundering-investigations-for-fiscal-year-2017>.

ACKNOWLEDGMENTS

The authors would like to thank David Stewart for his guidance and thought leadership regarding AML compliance. In addition, we would like to thank Adam Pilz for his guidance and thought leadership regarding text analytics.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Austin Cook
100 SAS Campus Drive
Cary, NC 27513

SAS Institute Inc.
Austin.Cook@sas.com
<http://www.sas.com>

Beth Herron
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Beth.Herron@sas.com
<http://www.sas.com>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Invoiced: Using SAS® Contextual Analysis to Calculate Final Weighted Average Consumer Price

Alexandre Carvalho, SAS Institute Inc.

ABSTRACT

SAS® Contextual Analysis brings advantages to the analysis of the millions of Electronic Tax Invoices (Nota Fiscal Eletrônica) issued by industries and improves the validation of taxes applied. Tax calculation is one of the analytical challenges for government finance secretaries in Brazil. This paper highlights two items of interest in the public sector: tax collection efficiency and the calculation of the final weighted average consumer price. The features in SAS® Contextual Analysis enable the implementation of a tax taxonomy that analyzes the contents of invoices, automatically categorizes the product, and calculates a reference value of the prices charged in the market. The first use case is an analysis of compliance between the official tax rate—as specified by the Mercosul Common Nomenclature (NCM)—and the description on the electronic invoice. (The NCM code was adopted in January 1995 by Argentina, Brazil, Paraguay, and Uruguay for product classification.) The second use case is the calculation of the final weighted average consumer price (PMPF). Generally, this calculation is done through sampling performed by public agencies. The benefits of a solution such as SAS Contextual Analysis are automatic categorization of all invoices and NCM code validation. The text analysis and the generated results contribute to tax collection efficiency and result in a more adequate reference value for use in the calculation of taxes on the circulation of goods and services.

INTRODUCTION

This paper focuses on the analytical challenges of government finance secretaries in Brazil, including the following:

- categorize the contents of the Electronic Tax Invoices
- improve the accuracy of the calculation of the final weighted average consumer price
- build an analytical base table that can be used as the basis for the calculation of the final weighted average consumer price

Business analysts and IT professionals are looking for solutions that are easy to use and easy to integrate into their existing systems, and that improve their analytics and their outcomes to challenges. SAS Contextual Analysis has benefits that combine machine learning and text mining with linguistic rules.

Some of these features can be directly monetized to help provide a fast return, such as the following:

- filtering documents
- predefined concepts
- ability to create and improving rules to concepts and categories
- exploring for new topics
- categorizing unstructured textual data and collections of documents

These and other features are found in SAS Contextual Analysis through a single integrated system. You can update and customize rules as needed.

DATA SOURCES FOR THIS DEMO

The data source was provided by and its use authorized by Secretaria de Estado de Fazenda de Minas Gerais (SEFA MG), Brazil. In May 2017, the data source was utilized in Proof of Concept (POC) for categorizing invoice issues. The results were reduced classification time, improved accuracy in product identification, and help with identifying anomalies in invoices and taxes.

Display 1 shows a sample of the data source with 9,955 rows and 6 variables (including descriptive text about the invoices and the NCM code). The sample contains grouped information about Electronic Tax Invoices issued to taxpayers (that is, industries). The Electronic Tax Invoices issued are a selection of the invoices issued in May 2017, and the source does not contain confidential information about taxpayers.

	DESCRIPTION_INVOICES	NCM CHAPTER	NCM POSITION	NCM SUB POSITION	NCM ITEM	NCM SUB ITEM
1362	BAVARIA LATA 350ML/12	22	2203	220300	2203000	22030000
1363	ANTARCTICA SUBZERO LATA 350ML SH ...	22	2203	220300	2203000	22030000
1364	BRAHMA CHOPP LT 473ML SH C 12 NPAL	22	2203	220300	2203000	22030000
1365	BRAHMA EXTRA LONG NECK 355ML SIX... ...	22	2203	220300	2203000	22030000
1366	SKOL LATA 350ML SH C/12 NPAL	22	2203	220300	2203000	22030000
1367	ORIGINAL 600ML	60.7915	2203	220300	2203000	22030000
1368	HEINEKEN VNR 355ML 1X1	22	2203	220300	2203000	22030000
1369	MILLER LN 355ML	22	2203	220300	2203000	22030000
1370	MALZBIER BRAHMA LONG NECK 355ML S...	22	2203	220300	2203000	22030000
1371	CERV SCHIN PILS 0.269LT 15 UN PBR	22	2203	220300	2203000	22030000
1372	BRAHMA CHOPP GFA VD 300ML CX C/23 ...	22	2203	220300	2203000	22030000
1373	KAISEL PILSEN LATA 350 ML	22	2203	220300	2203000	22030000
1374	SKOL LATA 350ML SH C 12 NPAL	22	2203	220300	2203000	22030000
1375	BUDWEISER LN 343ML SIXPACK CARTAO...	22	2203	220300	2203000	22030000
1376	0101 - CERVEJA NOVA SCHIN 600ML	22	2203	220300	2203000	22030003
1377	CHAMP CHANDON 187ML BABY BRUT RO...	22	2204	220410	2204101	22041010
1378	ESPUMANTE	22	2204	220410	2204101	22041010
1379	CHAMP CHUVA PRATA BRANCO 660ML	22	2204	220410	2204101	22041010
1380	VINHO NAC PERGOLA 1L TINTO SUAVE	22	2204	220410	2204101	22041010
...

Display 1. Data Source from SEFA-MG, 2017

UNDERSTANDING THE ICMS TAX AND THE CONTENT OF THE INVOICE DESCRIPTIONS

ICMS is the tax levied on the circulation of products such as food, beverages, household appliances, communication services, transportation, and some imported products, and became law in 1997 (also known as the Lei Kandir law). In Brazil, ICMS is one of the largest sources of financial revenue. Because it is established by each state (for example, Minas Gerais, Rio de Janeiro, or São Paulo), it changes from one place to another. Tax collections can be routed to various functions (for example, health, education, payment of civil servants, and so on).

At each stage of the collection cycle, it is always necessary to issue an invoice or tax coupon, which is calculated by the taxpayer and collected by the State. There are two types of Electronic Tax Invoices: invoices issued at the industry level (electronic invoices issued by the beer, refrigerator, or fuel industries) and invoices issued at the consumer level (electronic invoices issued by restaurants to final consumers).

In Display 2, line 1375 (BUDWEISER LN 343ML SIXPACK CARTAO) provides us with the following information: Product (Budweiser), Type (LN means Long Neck), Volume (343ML), and Quantity (SIXPACK CARTAO SH C/4).

	DESCRIPTION_INVOICES
1373	KAISEL PILSEN LATA 350 ML
1374	SKOL LATA 350ML SH C 12 NPAL
1375	BUDWEISER LN 343ML SIXPACK CARTAO SH C/4 68,1700
1376	0101 - CERVEJA NOVA SCHIN 600ML
1377	CHAMP CHANDON 187ML BABY BRUT ROSE(E)

Display 2. Data Source Content

WHAT IS THE MERCOSUL COMMON NOMENCLATURE (NCM CODE) FOR PRODUCT CLASSIFICATION?

The classification system for invoices follows the Mercosul Common Nomenclature (Nomenclatura Comum do Mercosul, or NCM) and was adopted in January 1995 by Argentina, Brazil, Paraguay, and Uruguay for product classification. Any merchandise, imported or purchased in Brazil, must have an NCM code in its legal documentation (invoices, legal books, and so on), whose objective is to classify the items according to the Mercosul regulation.

Display 3 shows examples of the content of Electronic Tax Invoices according to the NCM code by chapter, position, sub-position, item, and sub-item.

	DESCRIPTION_INVOICES	NCM CHAPTER	NCM POSITION	NCM SUB POSITION	NCM ITEM	NCM SUB ITEM
1373	KAISER PILSEN LATA 350 ML	22	2203	220300	2203000	22030000
1374	SKOL LATA 350ML SH C 12 NPAL	22	2203	220300	2203000	22030000
1375	BUDWEISER LN 343ML SIXPACK CARTAO SH C/4 68,1700	22	2203	220300	2203000	22030000
1376	0101 - CERVEJA NOVA SCHIN 600ML	22	2203	220300	2203000	22030003
1377	CHAMP CHANDON 187ML BABY BRUT ROSE(E)	22	2204	220410	2204101	22041010

Display 3. Mercosul Common Nomenclature Content

IMPROVING CATEGORIZATION EFFICIENCY WITH SAS CONTEXTUAL ANALYSIS

The use of unstructured data is growing exponentially in government agencies. In January 2018, according to the Brazilian Federal Revenue Agency (Receita Federal Brasileira), approximately 18 billion Electronic Tax Invoices were identified, and the number of issuers was approximately 1.4 million.

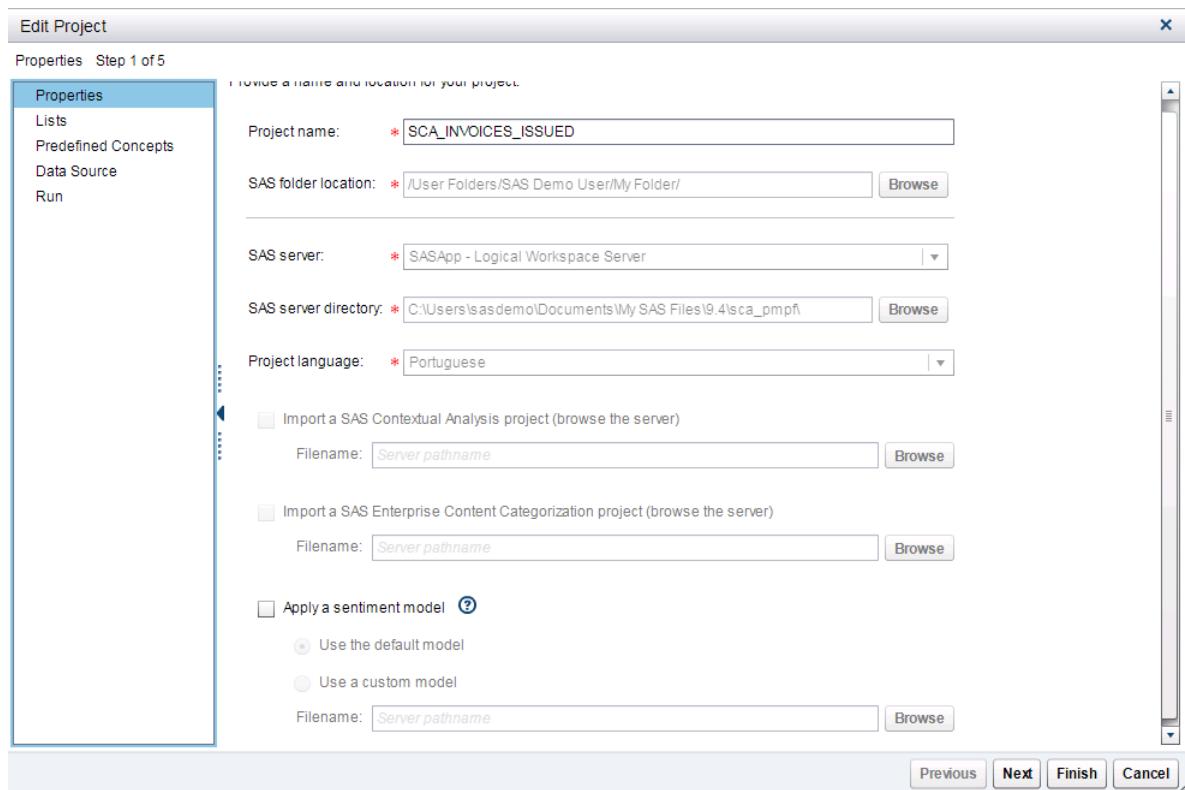
THE BENEFITS OF USING SAS CONTEXTUAL ANALYSIS

Business analysts are looking for solutions that are fast, easy to use and integrate into existing systems, and that improve their analytics and challenges. For the classification of electronic invoices, the analyst has more control with a hybrid approach. Analysts can add concepts (for example, 1LT, 500GR means quantity) and synonyms (skol, Budweiser, heinecken, brhma means beer) that specifically identify the product and its value for the tax aliquot calculation (for example, beer and 1LT the tax aliquot is 4%).

SAS Contextual Analysis combines machine learning and text mining capabilities with the ability to impose linguistic rules. SAS Contextual Analysis also enables you to filter, explore, and categorize unstructured textual data and collections of documents. Technology syntactically identifies common themes, category rules, and document sentiment, based on data. At any time, you can review and modify the results to meet your specific needs.

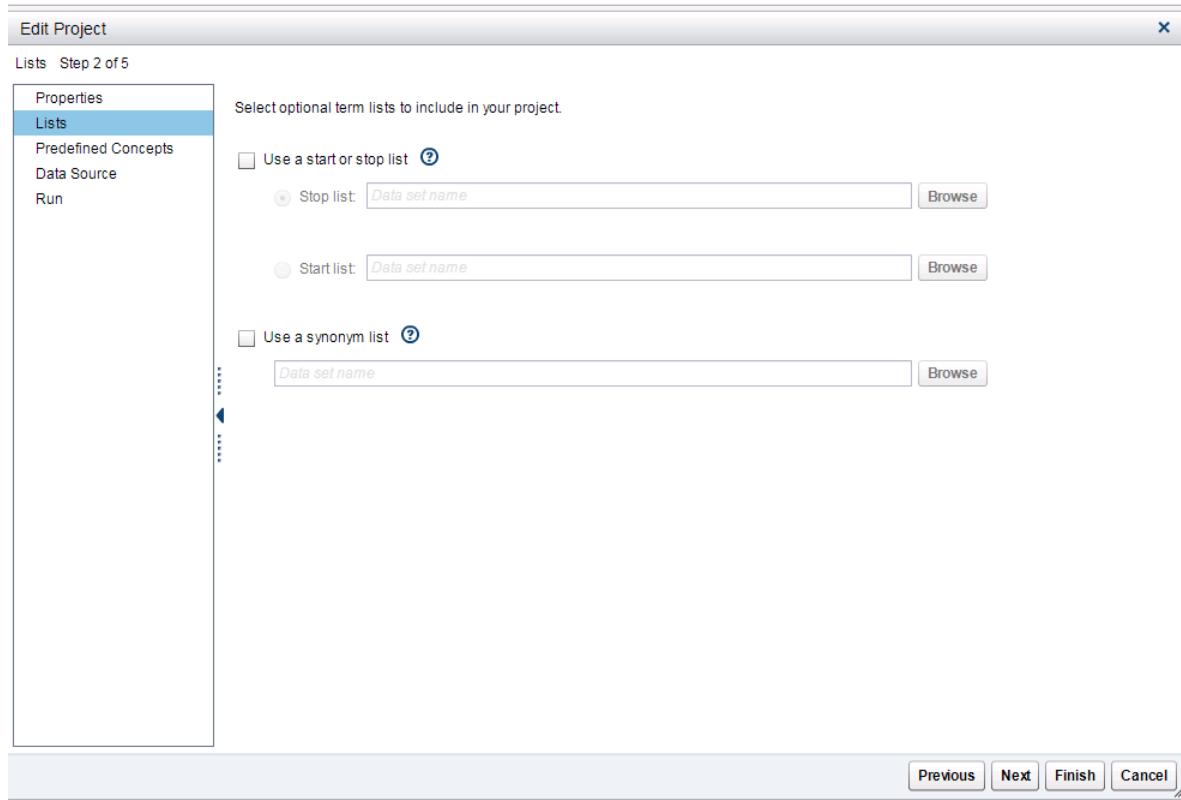
HOW TO BUILD A PROJECT IN SAS CONTEXTUAL ANALYSIS

Display 4 shows Step 1 of 5 for building a project in SAS Contextual Analysis. The analyst defines the name and location for your project, and chooses a project language. This paper doesn't apply a sentiment model, but is possible to use either the default model or a custom model.



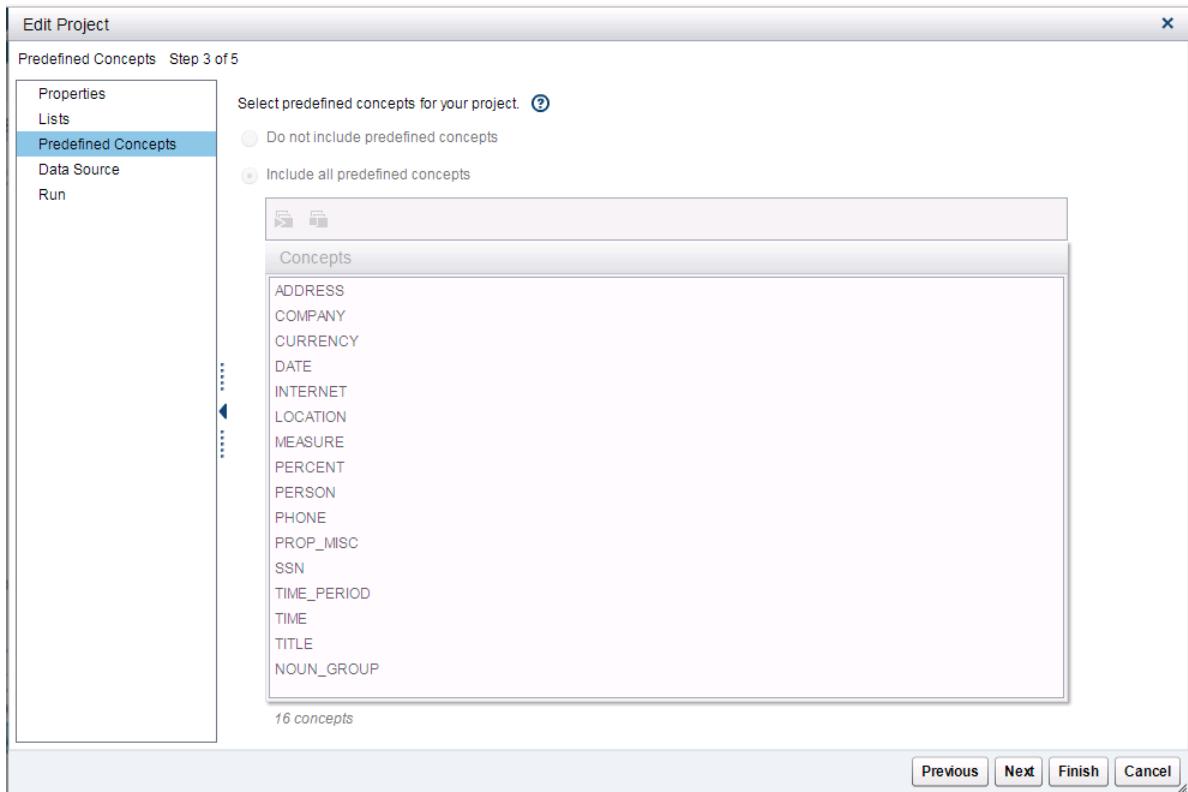
Display 4. Create a New Project: Define name, location and language for your project

Display 5 shows Step 2 of 5 for building a project in SAS Contextual Analysis. When analyzing text, it is common to disregard some terms already known to analysts that would not add value to the analysis or select a list of terms for research. For example, we can use the stop list (for name Brazil, SEFA-MG) or start list (skol, brahma, or budweiser). Another important feature is to use a list of synonyms whose terms would have the same meaning across the business (LT, GR, and KG all indicate quantity).



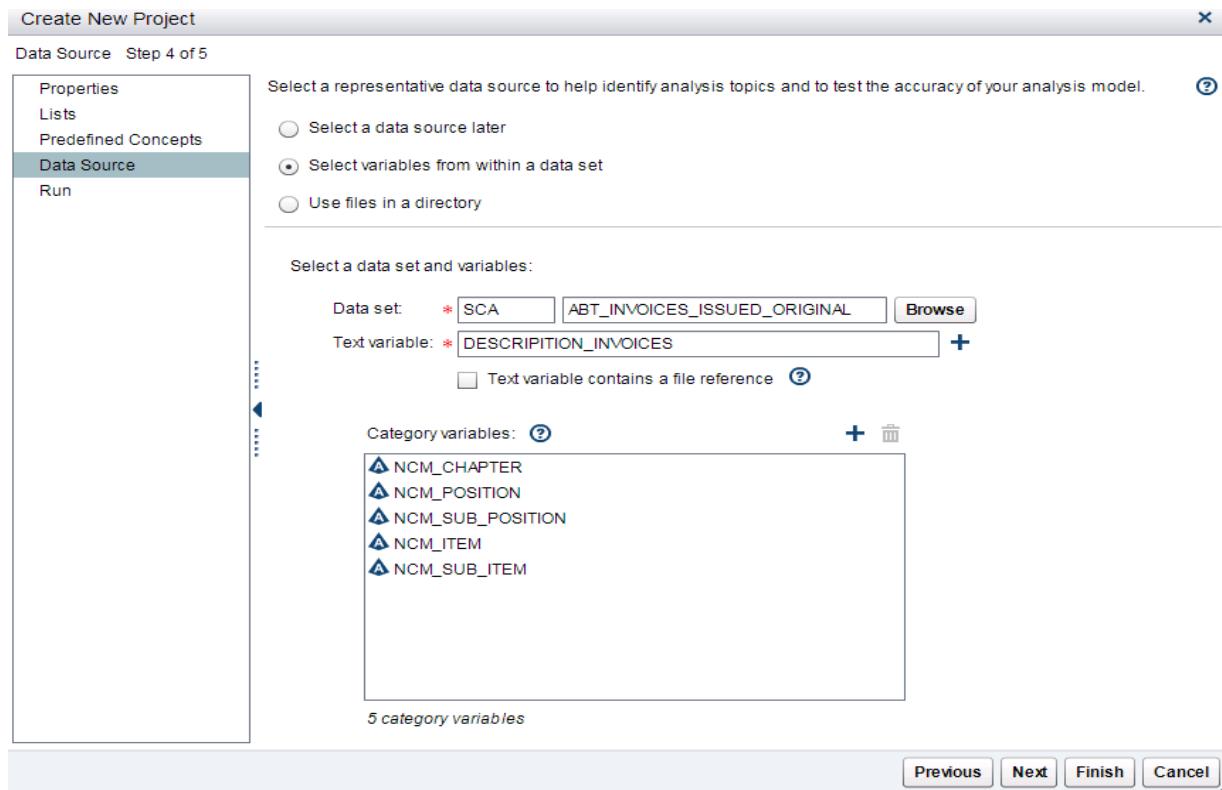
Display 5. Create a New Project: Define start list, stop list or synonyms list

Display 6 shows predefined concepts for your analysis and how SAS Contextual Analysis automatically identifies concepts such as location, currency, company, address, and so on.



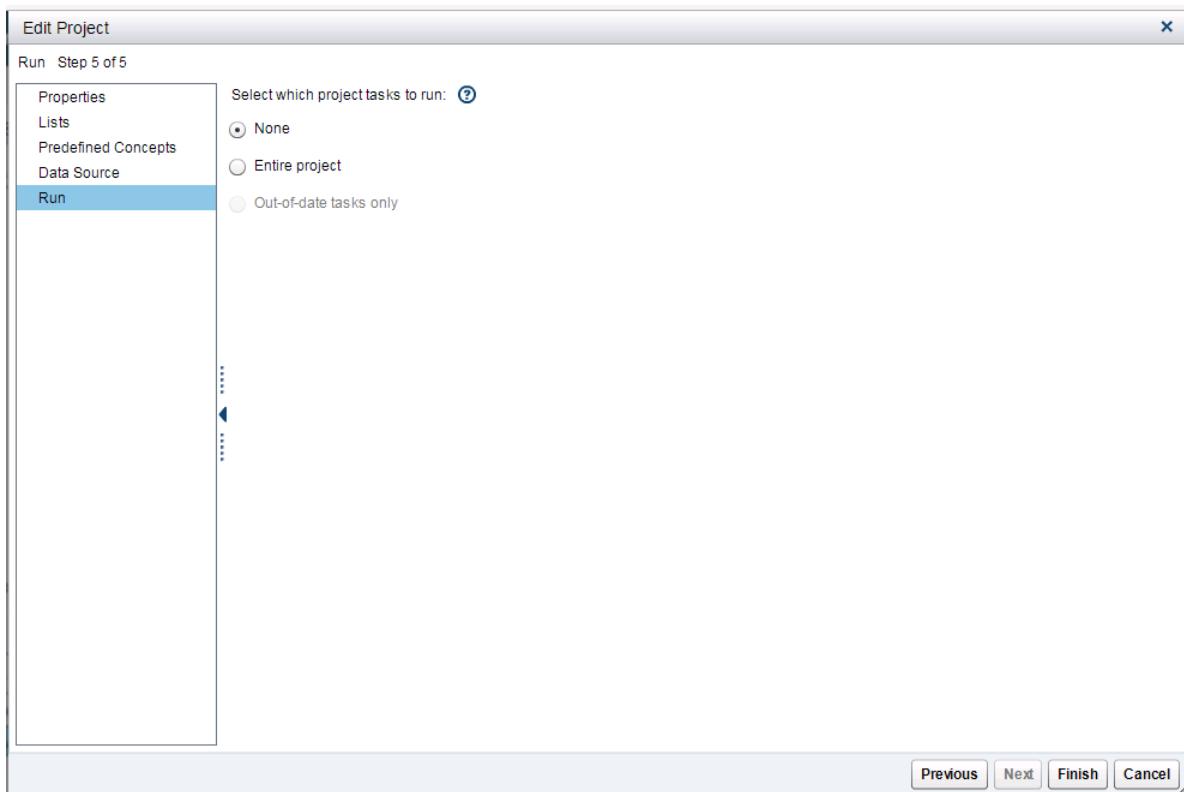
Display 6. Create a New Project: Predefined Concepts

Display 7 shows Step 4 of 5, which is when you select a SAS data set (ABT_INVOICES_ISSUED_ORIGINAL). The variable DESCRIPTION_INVOICES contains the invoice description, and text mining is used. On the other hand, NCM code information is used for categorization.



Display 7. Create a New Project: Select Data Set and Variables

And finally, you are ready to run the project (Display 8).



Display 8. Create a New Project: Run the entire project

IDENTIFY TERMS: NAME, TYPE, AND PRODUCT QUANTITY

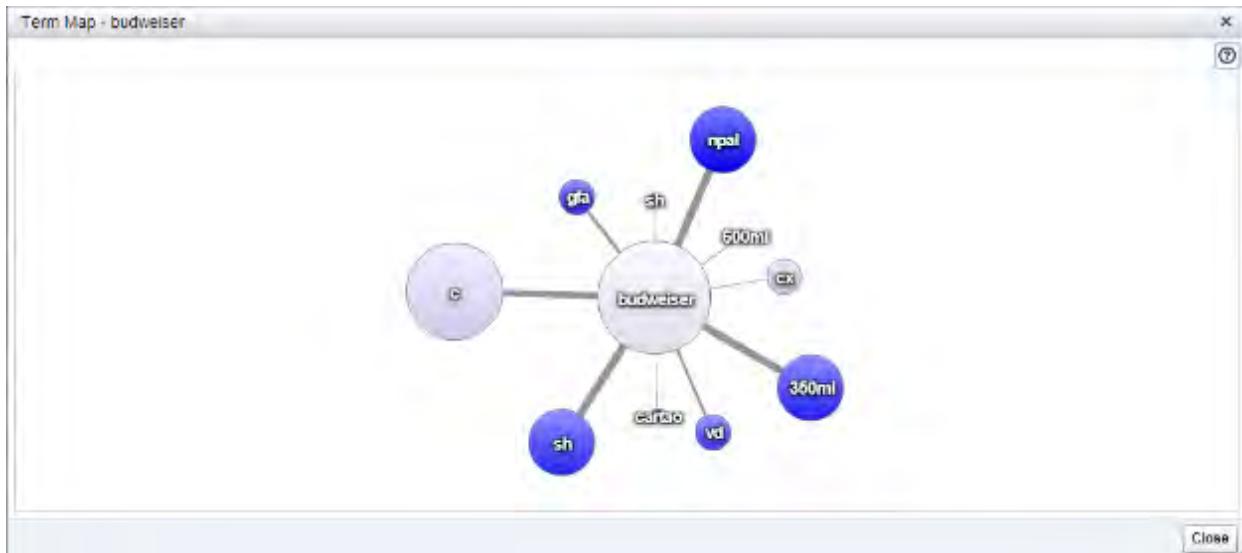
Display 9 focuses on the term "budweiser". In this case, you can see the stemming for the term "budweiser", including the three forms it takes and the few rare misspellings that have occurred in the documents (for example, "budwiser"). In this example, "budweiser" is the description of a type of beer (product name).

Terms and Synonyms	Number of Documents	Concept
bucho	93	
budweiser	9	PROP_MISC
budweiser	5	PROP_MISC
budweiser	3	
budwiser	1	PROP_MISC
bujao	10	
bull	3	
business	3	
bwg	3	PROP_MISC
by	3	
c	651	
c100	3	
c15	4	

Term > budweiser	Documents
BUDWEISER LATA 350ML SH C 12 NPAL	24,4200
BUDWEISER LATA 350ML SH C/12 NPAL	27,9600
BUDWEISER GFAVD 990ML CX C/12	68,2100

Display 9. Create a New Project: Identifying Terms

In the term map shown in Display 10, you can see that there is additional information about the product type (for example, "gf" and "cx" mean "bottle") and volume (350ml or 600ml). The term map can help you refine your terms list and create rules for classification.



Display 10. Create New Project: Term Map

DISCOVERING TOPICS FOR THE ELECTRONIC TAX INVOICES

In particular, the Topics functionality in SAS Contextual Analysis can help you to automatically identify the contents of your documents, which are in this case Electronic Tax Invoices.

Display 11 shows the documents for the topic **+lata+350ml,sh,+npal+brahma**. On the right side of the window, you can see a set of tax invoices that identify as a type of beer.

The screenshot shows the SAS Contextual Analysis interface with the following components:

- Top Bar:** SAS® Contextual Analysis, File, Help, Sign Out.
- Toolbar:** SCA_INVOICES_ISSUED, Topics, Run, View.
- Topics Tab:** A table titled 'Topics' showing document counts for various topics. One row is highlighted for '+lata,350ml,sh,+npal+brahma' with a count of 104.
- Documents Tab:** A table titled 'Documents' showing tax invoices for the topic '+lata,350ml,sh,+npal+brahma'. Three entries are listed:
 - BRAHMA CHOPP LATA 350ML SH C 12 NPAL 18,2168
 - BRAHMA CHOPP LATA 350ML SH C 12 NPAL 19,6200
 - BRAHMA CHOPP LATA 350ML SH C 12 NPAL 20,2200

Display 11. Identify Emerging Issues

In Display 12 and Display 13, you see the **Terms** tab, on which you can choose from two different views of the terms that constitute the topics. You can also choose different views of the documents that are associated with the topics.

The screenshot shows the SAS Contextual Analysis interface. The top navigation bar includes 'File' and 'Help' on the left, and 'Sign Out' on the right. Below the navigation is a toolbar with icons for 'Run', 'View', and file operations. The main area has a title 'SCA_INVOICES_ISSUED'. On the left, a 'Topics' panel lists various search terms with their document counts. On the right, a 'Terms' panel displays a list of terms with their weights, concepts, and document counts.

Topics

Topics	Number of Documents
+kit,+reparar,barra,+extintor,+mb	103
+lampada,fluor,127v+lampada,24v	217
+lata,350ml,sh,+npal,+brahma	104
+lubrificante,+blindado,filtro blindado,filtro lubrificante,+elemento	194
+luva,g,+latex,+correr,+mucambo	124
+mangueira,+vermelho,+flex,aco,+50mm	78
+ml,+limpar,+tintar,spray,+aromatizante	120
+motor,+juntar,+oleo,+lub,cx	144
+palhetar,universal,silicone,+flex,pc	176
+para,+com,+adesivo,+elemento,filtrante	175
+parafusar,+sex+,+auto,+atar,+panela	125
+pet,+guarana,+antarctica,2l,chp	101
+placar,+2x4,+suportar,+modular,+cegar	123
+pneu,+80r22,r,+limpar,82	72

Terms

Term	Weight	Concept	Number of Documents
lata	0.522	TYPE_BOTTLE	66
350ml	0.438	VOLUME_ML	37
sh	0.432	PROP_MISC	38
npal	0.411	PROP_MISC	38
brahma	0.178		26
chopp	0.177		23
skol	0.162	PROP_MISC	14
brahma chopp	0.114	PROP_MISC	9
antarctica	0.099		31
It	0.096		33
budweiser	0.081	PROP_MISC	9
473ml	0.079	VOLUME_ML	9

Display 12. The Terms Tab: View Tabular Form

SAS® Contextual Analysis

File Help Sign Out

SCA_INVOICES_ISSUED

Topics Run View

Topics

Topics	Number of Documents
+kit,+reparar,barra,+extintor,+mb	103
+lampada,fluor,127v,lampada,24v	217
+lata,350ml,sh,+npal,+brahma	104
+lubrificante,+blindado,filtro blindado,filtro lubrificante,+elemento	194
+luva,g,+latex,+correr,+mucambo	124
+manguera,+vermelho,+flex,aco,+50mm	78
+ml,+limpar,+tintar,spray,+aromatizante	120
+motor,+juntar,+oleo,+lub,cx	144
+palhatar,universal,silicone,+flex,pç	176
+para,+com,+adesivo,+elemento,filtrante	175
+parafusar,+sext,+auto,+atar,+panela	125
+pet,+guarana,+antarctica,2l,chk	101
+placar,+2x4,+suportar,+modular,+cegar	123
+pneu,+80x22,r+limpar,82l	72

Terms Documents

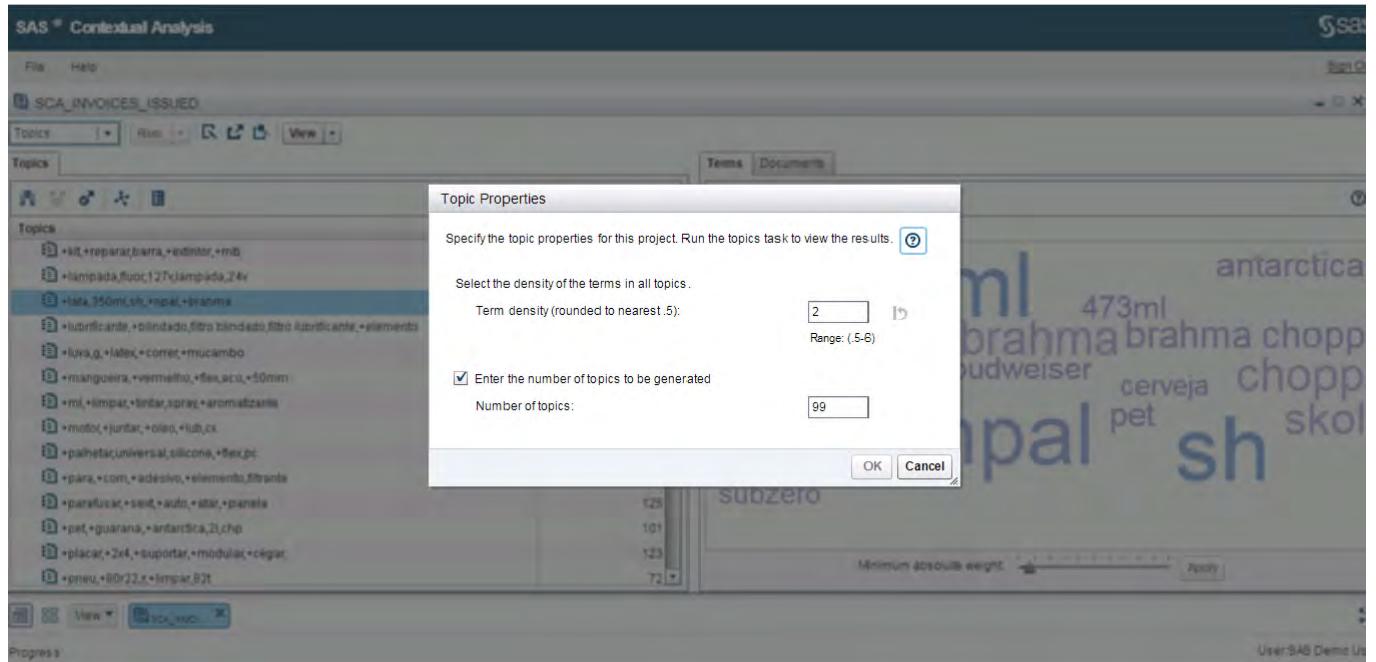
Topic > +lata,350ml,sh,+npal,+brahma

A word cloud visualization showing the frequency of terms related to the topic. The most prominent words are 'lata', '350ml', 'sh', 'npal', 'brahma', and 'chopp'. Other visible words include 'subzero', 'anarctica', '473ml', 'antarctica', 'cerveja', 'skol', 'pet', 'budweiser', 'chopp', 'brahma', 'chopp', 'lt', 'it', 'subzero', and 'anarctica'.

Minimum absolute weight: 2 Apply

Display 13. The Terms Tab: View Graphic Form

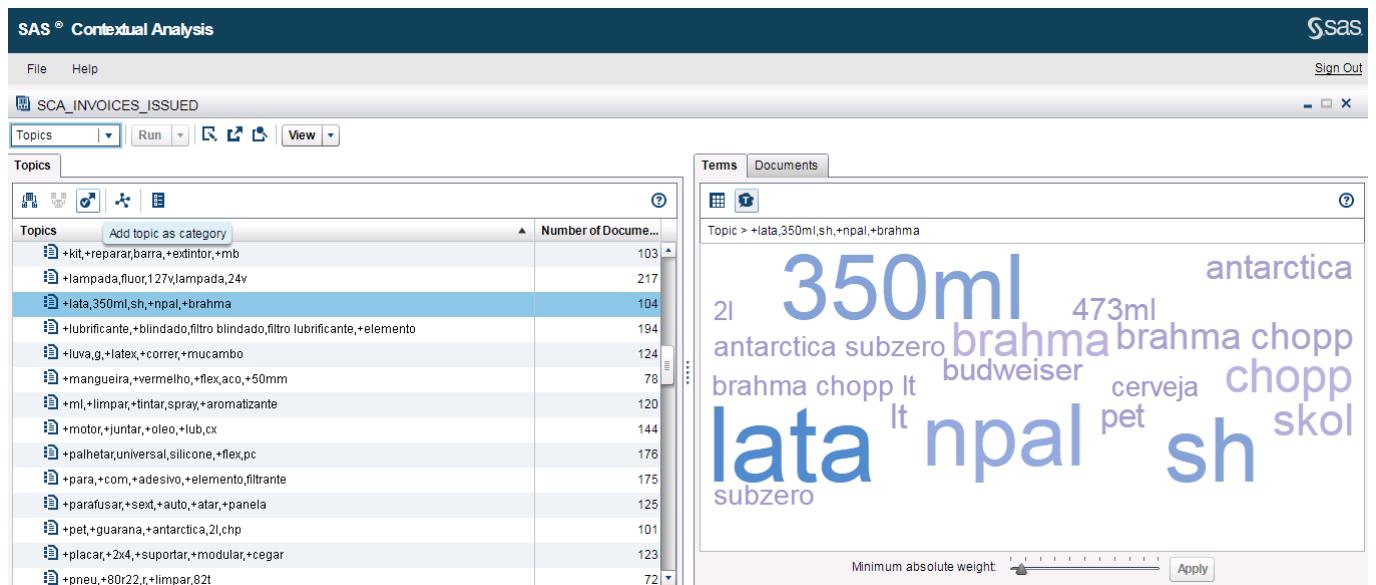
In some situations, the analyst needs to define a specific number of topics because of the structure of their challenges. In Display 14, we change the number of the topics to 99.



Display 14. Topic Properties

HOW TO TRANSFORM TOPICS INTO CATEGORIES

After the topics are validated, you can create categories. Let's continue with the topic that identifies drinks, and promote some topics to be categories. First, you choose a topic and click the Add Topic icon, as shown in Display 15.



Display 15. Promote Topics to Categories

SAS Contextual Analysis suggests possible rules for classifying newly issued invoices. In this example, we transform the topic, which is the type of drinks, into a category that is defined as BEERS (see Display 16). On the **Documents** tab, you can see that out of 9,955 documents, 108 were categorized belonging to the BEERS category.

This analysis evaluates how well the displayed linguistic definitions of the categories approximate the underlying machine learning definitions. This is important because you will use the linguistic definitions to score other documents.

The screenshot shows the SAS® Contextual Analysis interface. The left pane displays the 'Categories' tab, which lists various document categories with their document proportions and counts. The right pane displays the 'Documents' tab, which lists individual documents with their IDs, descriptions, and counts. A red box highlights the 'Edit Rules' tab at the top of the right pane.

ID	Text	Count
1300	BRAHMA CHOPP GFAVD 300ML CX C/23	31,6036
1301	BRAHMA CHOPP GFAVD 300ML CX C/23	25,3000
1303	BUDWEISER LATA 350ML SH C/12 NPAL	27,9600
1305	ANTARCTICA SUBZERO LATA 350ML SH C 12 NPAL	14,6724
1306	BRAHMA CHOPP LT 473ML SH C/12 NPAL	22,5888
1308	CERVEJA SKOL LATA 350ML	
1309	BRAHMA CHOPP GFAVD 300ML CX C/23	29,9000
1310	KAI SER PILSEN LATA 473ML 12X1FT	
1311	BRAHMA CHOPP GFAVD 300ML CX C/23	31,1470
1312	BRAHMA CHOPP LT 473ML SH C/12 NPAL	26,7300
1313	BRAHMA CHOPP GFAVD 1L COM TT C	44,4240
1314	SKOL LT 269ML SH C15 NPAL	28,8000
1316	SKOL LATA 350ML SH C/12 NPAL	23,9467
1317	BRAHMA CHOPP 300ML	14,3200

Display 16. Examples of a Category and Its Taxonomies

CATEGORIZATION: EDIT RULES AND SCORE NEW DOCUMENTS

One of the first challenges for the business analyst is to develop a taxonomy that automatically categorizes invoice issues and that is updated in a recurring and more accurate manner according to the NCM. The results and benefits of accomplishing this are immediate, such as properly identifying the tax rate (for example, ICMS) and identifying possible anomalies in the application of the tax rate.

Display 17 shows the new category available in the Categories section. At this point, the analyst can improve the categorization process with the inclusion of his business knowledge on the **Edit Rules** tab.

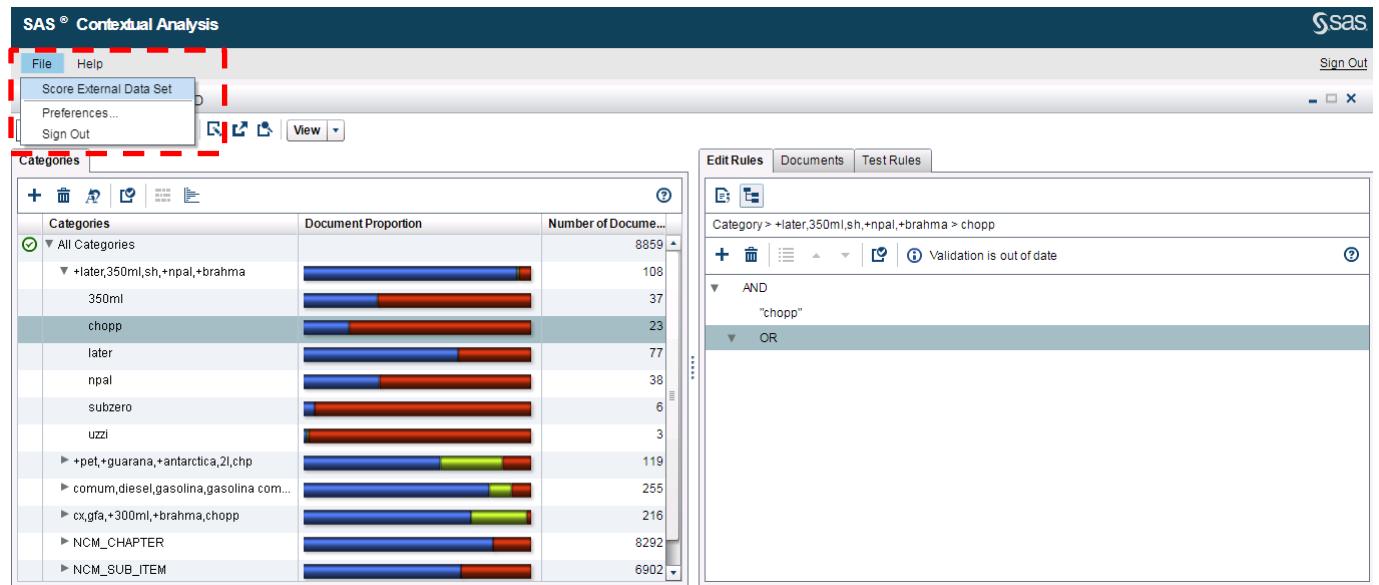
The screenshot shows the SAS® Contextual Analysis interface. The left pane displays the 'Categories' tab, listing document categories. The right pane displays the 'Edit Rules' tab, which shows a hierarchical rule definition. A red box highlights the rule definition area.

```

Category > +later,350ml,sh,+npal,+brahma > chopp
+ AND
  "chopp"
    OR
      NOTSENT
    OR
      ORD
      ORDDIST_
      PAR
  
```

Display 17. Examples of Categories and Their Taxonomies

You can also use models developed in SAS Contextual Analysis to score additional text data. Select **File>Score External Data Set** (see Display 18). A window appears in which you can identify the textual data that you want to score. Additionally, you can view and export the DS2 macro code used to define concepts, sentiment, and categories for use anywhere you can run SAS.



Display 18. Score External Data Set

Display 19 shows the results after categorization. The variable *document_id* is the ID of the invoices; the variable *name* is the name of the category, and the text with the description of the notes is in the *Description_Invoices* column.

	document_id	DESCRIPTION_INVOICES	name	term	column_na ...
49	1278	FUSION LATA 250ML SIX-PACK	Top/+later,350ml,sh,+npal,+brahma	LATA	c_994
50	1284	BUDWEISER LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	350ML	c_994
51	1284	BUDWEISER LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	LATA	c_994
52	1284	BUDWEISER LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	NPAL	c_994
53	1285	BRAHMA CHOPP LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	350ML	c_994
54	1285	BRAHMA CHOPP LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	CHOPP	c_994
55	1285	BRAHMA CHOPP LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	LATA	c_994
56	1285	BRAHMA CHOPP LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	NPAL	c_994
57	1287	SKOL LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	350ML	c_994
58	1287	SKOL LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	LATA	c_994
59	1287	SKOL LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	NPAL	c_994

Display 19. Categorization Result

INPUTS FOR CALCULATING THE FINAL WEIGHTED AVERAGE CONSUMER PRICE

The calculation of the final weighted consumer average price is updated frequently, and the values for some products rise more than others. In Brazil, the most common products for which the ICMS is calculated based on the final weighted average consumer price are fuels, drinks, and cosmetics, among other goods.

The taxpayer needs to be aware of this calculation and determine whether they are subject to it. Otherwise, taxpayers might end up doing their ICMS calculations erroneously.

For this reason, there is a need to extract concepts like volume, type, quantity, and product name from the thousands or millions of Electronic Tax Invoices for inclusion in the calculation of the final weighted average consumer price.

HOW SAS CONTEXTUAL ANALYSIS ENRICHES THE CALCULATION

SAS Contextual Analysis uses language interpretation and text interpretation (LITI) syntax and its concept rules to recognize terms like. kg, ml, bottle, and so on, in context, so that you can extract only concepts in a document (for example, "Budweiser 355ML") that match your rule.

In Display 20, you can see a custom concept node named VOLUME_LT and regular expressions (Regex syntax). These elements will extract all Electronic Tax Invoices in our data source that contain "LT" and all combinations that include numbers (RECEX: [0-9]*LT). The operator - is a wildcard that matches any character.

The screenshot shows the SAS Contextual Analysis interface. On the left, the 'Concepts' pane lists predefined and custom concepts. Under 'Custom Concepts', 'VOLUME_LT' is selected. On the right, the 'Edit Rules' pane displays the rules for this concept:

```
Custom Concept > VOLUME_LT
1 REGEX: [0-9]*ML
2 REGEX: [0-9]*LT
3 REGEX: [0-9]*L
4
5
6
7
8
9
10
11
12
13
14
15
16
```

Display 20. Custom Concepts and Editing Rules for VOLUME_LT

Display 21 shows the rule for identifying all Electronic Tax Invoices for the concept node TYPE_BOTTLE that contain the terms "GFA, LATA, GARRAFA" and any number combination. Each document is evaluated separately for matches (shown in Display 22).

The screenshot shows the SAS Contextual Analysis interface. On the left, the 'Concepts' pane lists predefined and custom concepts. Under 'Custom Concepts', 'TYPE_BOTTLE' is selected. On the right, the 'Edit Rules' pane displays the rules for this concept:

```
Custom Concept > TYPE_BOTTLE
1 REGEX: [0-9]*GFA
2 REGEX: [0-9]*LATA
3 REGEX: GFA
4 REGEX: GARRAFA
5 REGEX: [0-9]*GFA
6 REGEX: [0-9]*GARRAFA
7
8
9
10
11
12
13
14
15
16
```

Display 21. Custom Concepts and Editing Rules for TYPE_BOTTLE

The screenshot shows the SAS Contextual Analysis interface. On the left, the 'Concepts' pane displays a tree structure with 'Predefined Concepts (16)' and 'Custom Concepts (5)'. Under 'Custom Concepts', there are five entries: BEER, OIL, TYPE_BOTTLE, VOLUME_KG, and VOLUME_DL. The 'TYPE_BOTTLE' node is selected. On the right, the 'Documents' pane shows a list of 66 documents from a custom concept named 'TYPE_BOTTLE'. The list includes various product names and their details, such as 'LATA SONHO DE VALSA 236G', 'MILHO VERDE QUERO LATA 200G', and 'GUARANA LATA'. The document IDs range from 585 to 1240.

Display 22. Results of the Custom Rule for TYPE_BOTTLE

ANALYTICAL BASE TABLE FOR THE CALCULATION

Today, the final weighted average consumer price is typically obtained from sample surveys of final consumer prices. Such surveys can be ordered from the Finance Secretary.

Display 23 shows an example created in the SAS Contextual Analysis, which shows a possible analytical basis that can be used in the final weighted final consumer price calculation. The variable *document_id* represents the identification of the electronic invoice, *DESCRIPTION_INVOICES* contains the contents of the invoice, *name* is the category, and *term* is the result of extracting the electronic invoice concepts.

As an example, we could calculate the average final consumer price of all invoices classified as BEERS (*name*) and sold in cans of 355ML (*term* = "can" and *name* = "+ chopp + brhma + ..."). This process would already be automated, and it would be possible to generate reports in SAS Visual Analytics. This same logic would enrich the calculation for other items like food, building materials, and so on.

	document_id	DESCRIPTION_INVOICES	name	term
1	585	LATA SONHO DE VALSA 236G	Top/+later,350ml,sh,+npal,+brahma	LATA
2	746	PANETT BAUDUCCO LATA DOURADA 1KG	Top/+later,350ml,sh,+npal,+brahma	LATA
3	888	MILHO VERDE QUERO LATA 200G	Top/+later,350ml,sh,+npal,+brahma	LATA
4	1069	GUARANA LATA	Top/+later,350ml,sh,+npal,+brahma	LATA
5	1142	CITRUS ANTARCTICA LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	350ML
6	1142	CITRUS ANTARCTICA LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	LATA
7	1142	CITRUS ANTARCTICA LATA 350ML SH C 12 NPAL	Top/+later,350ml,sh,+npal,+brahma	NPAL
8	1144	SODA LIMONADA ANTARCTICA LATA 350ML SH C/1 1...	Top/+later,350ml,sh,+npal,+brahma	350ML
9	1144	SODA LIMONADA ANTARCTICA LATA 350ML SH C/1 1...	Top/+later,350ml,sh,+npal,+brahma	LATA
10	1150	COCA COLA LATA ZERO 350 ML	Top/+later,350ml,sh,+npal,+brahma	LATA
11	1160	GUARANA CHP ANTARCTICA DIET LATA 350ML SH 1...	Top/+later,350ml,sh,+npal,+brahma	350ML
12	1160	GUARANA CHP ANTARCTICA DIET LATA 350ML SH 1...	Top/+later,350ml,sh,+npal,+brahma	LATA

Display 23. Calculating the Final Weighted Average Consumer Price

CONCLUSION

This paper shows how you can use SAS Contextual Analysis to automate the process of product categorization and create custom concepts, using data that supports the calculation of the tax for the Electronic Tax Invoice. This methodology can be used in other Mercosul countries to reduce analysis

time. This methodology can also improve governance, trust, and accuracy for the validation of invoice issues.

REFERENCES

COSTA, Leonardo De Andrade. 2015. "Processo Administrativo Tributário." FGV Direito Rio. Available https://direitorio.fgv.br/sites/direitorio.fgv.br/files/u100/processo_administrativo_tributario_2015-2.pdf

SAS Contextual Analysis 14.1: Reference Help.

Website "O seu Portal Jurídico da internet-Âmbito Jurídico". Available <http://www.ambito-juridico.com.br/site/>. Accessed on February 20, 2018.

Website "Portal da Nota Fiscal Electrônica". Available <http://www.nfe.fazenda.gov.br/portal/principal.aspx>. Accessed on February 20, 2018.

ACKNOWLEDGMENTS

The author thanks Mauricio Fonseca Fernandino and Secretaria de Fazenda de Minas Gerais for Data Sources. The author thanks Joan Keyser and Amy Wolfe for help in reviewing the text.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Alexandre Carvalho
SAS Institute Brasil
55 21 99121-3280
Alexandre.carvalho@sas.com
<https://br.linkedin.com/in/alexandrecarvalho>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Using SAS® Text Analytics to Assess International Human Trafficking Patterns

Tom Sabo, Adam Pilz, SAS Institute Inc

ABSTRACT

The US Department of State (DOS) and other humanitarian agencies have a vested interest in assessing and preventing human trafficking in its many forms. A subdivision within the DOS releases publicly facing Trafficking in Persons (TIP) reports for approximately 200 countries annually. These reports are entirely freeform text, though there is a richness of structure hidden within the text. How can decision-makers quickly tap this information for patterns in international human trafficking?

This paper showcases a strategy of applying SAS® Text Analytics to explore the TIP reports and apply new layers of structured information. Specifically, we identify common themes across the reports, use topic analysis to identify a structural similarity across reports, identifying source and destination countries involved in trafficking, and use a rule-building approach to extract these relationships from freeform text. We subsequently depict these trafficking relationships across multiple countries in SAS® Visual Analytics, using a geographic network diagram that covers the types of trafficking as well as whether the countries involved are invested in addressing the problem. This ultimately provides decision-makers with big-picture information about how to best combat human trafficking internationally.

INTRODUCTION

Human trafficking is one of the most tragic human rights issues of our time. It splinters families, distorts global markets, undermines the rule of law, and spurs other transnational criminal activity. It threatens public safety and national security¹. The International Labour Organization estimates that there are 20.9 million victims of human trafficking globally, and that forced labor and human trafficking generates 150 billion dollars in illicit gains annually. Of the 20.9 million victims, 26% are children, and 55% are women and girls².

The U.S. Department of state produces the Trafficking in Persons (TIP) report annually. It assesses the state of human trafficking in approximately 200 countries. This report is the U.S. Government's principal diplomatic tool to engage foreign governments on human trafficking. It is also the world's most comprehensive resource of governmental anti-trafficking efforts and reflects the U.S. Government's commitment to global leadership on this key human rights and law enforcement issue. It is used by the U.S. Government and worldwide as a tool to engage in dialogs to advance anti-trafficking reforms, and examine where resources are most needed. Freeing victims, preventing trafficking, and bringing traffickers to justice are the ultimate goals of the report and of the U.S Government's anti-trafficking policy¹. However, the insights in these reports are scattered across hundreds of free-form text documents. How can we make the data in these reports more accessible to the broad audience that it supports, and how can we better envision patterns in the data which can be used to combat human trafficking?

This paper showcases a combination of SAS technology to identify patterns in the reports ultimately making the information more accessible to the various stakeholders. In particular, we will show how SAS can be used to identify links between source and destination countries, and visually depict these geospatial patterns in a network diagram. In this process, we will apply text analytics and visualization capabilities, primarily from SAS Visual Text Analytics and SAS Visual Analytics, available through SAS Viya. We will answer the following questions.

- Can we assess overall themes in international trafficking from the reports?
- Can we identify more focused patterns in trafficking, such as how women and children are seeking and achieving refuge?
- Can we identify and geospatially visualize patterns in trafficking across countries, including who is being trafficked (men, women, children), what type of trafficking is occurring (labor or sex trafficking), and whether the countries in question are in cooperation to address the problem?

By the end of this paper, the reader will learn how to apply the full SAS analytics lifecycle to this problem³. In this case, the analytics lifecycle includes data acquisition, unstructured and structured data management, text analytics, and network visualization. The reader will also gain an understanding of some key features available in SAS Visual Text Analytics, including similarity scores and fact extraction. We will also highlight some functionality common to the SAS text analytics products, including capabilities available across SAS Visual Text Analytics, SAS Contextual Analysis, and SAS Text Miner.

DATA ACQUISITION AND DATA MANAGEMENT

We obtained the data for each country narrative from the U.S. Department of State Trafficking in Persons report for 2017 using a script that accessed the following link:

<https://www.state.gov/i/tip/rls/tip rpt/countries/2017/index.htm>. A slight modification to the script enabled us to obtain country narrative data from 2013-2016. Each report contains summary information about trafficking in the country, as well as several subsections. Subsections include recommendations, how the country prosecutes human traffickers, how the country protects victims, how the country prevents trafficking, and an overall trafficking profile.

The country level trafficking reports are several pages in length. When working with documents greater than a page or two, it is helpful to apply some level of tokenization prior to text analytics⁴. Longer documents are more likely to have multiple themes embedded within. Tokenization breaks the documents up into smaller chunks, while maintaining a reference for each chunk to the larger documents. Then, patterns that appear across chunks can be readily surfaced using the capabilities at our disposal. This makes our algorithms more likely to identify these discrete themes or topics within documents.

For this effort, we applied sentence level tokenization. The following is SAS code we used for sentence level tokenization. In this case, it accepted as input a SAS data set that contained a number of rows of data, each containing a country level narrative from the TIP reports from 2013-2017.

```
/*Define the library where the data set is stored*/
libname _mylib 'D:\data\SamplePDF';

/*Define the data set for which you desire tokenized sentences*/
%let dsn = _mylib.output_sas_data;

/*Define the text variable to parse*/
%let text_var = text;

/*Strip the data and create an index*/
data temp (compress=yes); set &dsn;
  doc_id = _n_;
run;

/*parse the data set*/
proc hptmine data=temp;
  doc_id doc_id;
  var &text_var;
  parse
    nostemming notagging nonounwords shownumpunct
    entities = none
    outpos   = position
    buildindex ;
  performance details ;
run;
```

```

proc sort data=position;
  by document sentence _start_;
run;

data sentenceSize (compress=yes);
  retain document start size;
  set position;
  by document sentence;
  if First.sentence then start=_start_+1;
  if Last.sentence then do;
    size=_end_-start+2;
    output;
  end;
  keep document start size;
run;

/*Clean up*/
proc delete data=position; run;

data sentenceObs(compress=yes);
  length sentences $1000;
  merge sentenceSize(in=A) temp (rename=(doc_id=document));
  by document;
  if A then do;
    sentences=substrn(&text_var,start,size);
    output;
  end;
  keep sentences document;
run;

/*Clean up*/
proc delete data=sentenceSize; run;

data _mylib.output_sentences(compress=yes);
  set sentenceObs;
  by document;
  if first.document then sid = 1; else sid + 1;
run;

/*Clean up*/
proc delete data=sentenceObs; run;

/*view the data*/
proc print data=_mylib.output_sentences (obs=100);
run;

```

The output data set from the sentence tokenization procedure contained a row of data for each sentence in the original country level trafficking narratives, maintaining year, country, and sentence ID. This amounted to 63,648 rows of sentence level data. The following figure depicts a snapshot of the data. We took a 15,000 row sample of this sentence level data across all countries and years to use in the text analytics exercise described in the next section.

	year	country	sentence_id	sentence
51050	2017	Brunei	44	TRAFFICKING PROFILE.. As reported in the last five years, Brunei is a destinati...
51051	2017	Brunei	45	Men and women from Indonesia, Bangladesh, China, the Philippines, Thailand,...
51052	2017	Brunei	46	Upon arrival, some are subjected to involuntary servitude, debt bondage, non-pa...
51053	2017	Brunei	47	Some migrants who transit Brunei become victims of sex or labor trafficking upo...
51054	2017	Brunei	48	Some Bruneian women and girls are subjected to sex trafficking domestically.
51055	2017	Brunei	49	Although it is illegal for employers to withhold wages of domestic workers for mor...
51056	2017	Brunei	50	Retention of migrant workers travel documents by employers or agencies remain...
51057	2017	Bulgaria	1	Office To Monitor and Combat Trafficking in Persons...2017 Trafficking in Perso...
51058	2017	Bulgaria	2	The government demonstrated significant efforts during the reporting period by i...
51059	2017	Bulgaria	3	Law enforcement continued to take action against public officials and police offic...
51060	2017	Bulgaria	4	However, the government did not demonstrate increasing efforts compared to th...
51061	2017	Bulgaria	5	Although the total number of investigations and prosecutions of traffickers increa...
51062	2017	Bulgaria	6	The governments capacity to shelter victims and provide specialized services re...
51063	2017	Bulgaria	7	Because the government has devoted sufficient resources to a written plan that, i...
51064	2017	Bulgaria	8	Therefore, Bulgaria remained on Tier 2 Watch List for the third consecutive year.
51065	2017	Bulgaria	9	RECOMMENDATIONS FOR BULGARIA..Enhance efforts to investigate, prosec...

Figure 1: Sentence Level Country Narrative Data Used in Text Analytics

TEXT ANALYTICS

SAS now has a variety of capabilities in text analytics available in different solution packages. This includes capabilities in SAS Visual Text Analytics, available as a part of SAS Viya. This also includes capabilities present in SAS Text Miner, an add-on to SAS Enterprise Miner, and SAS Contextual Analysis, both available on any SAS 9 release. In this section on text analytics methods, we will show snapshots from individual solutions, and discuss which of the aforementioned SAS products also have the described capability.

IDENTIFYING OVERALL HUMAN TRAFFICKING TRENDS AND PATTERNS

One of the questions previously identified is whether we can assess overall themes in international trafficking from the reports. This is a capability available through an unsupervised machine learning method in text clustering. SAS assesses all the sentences across TIP reports and identifies key sets of terms that tend to occur together. For example, the terms “forced”, “child”, “beg”, and “street” tend to co-occur in the data along with other terms. These are indicative of a pattern across country narratives where children are coerced into begging. The following snapshot takes results from the text clustering capabilities of SAS Text Miner, and depicts the cluster results along with example sentences associated with the text cluster.

Documents Nearest to the Center of Cluster 1		# docs: cluster / corpus 249 / 3187 (7.8%)
Terms:	tokenized_sentence	Distance to Cluster Seed
<i>forced child beg street border roma koranic teacher mali school</i>	Some Ukrainian children are subjected to forced begging.	0.686142
	Forced begging was on the rise in 2012.	0.69465
	Children, particularly Roma, are subjected to forced begging.	0.741508
	Ethnic Roma children may be subjected to forced begging on the street.	0.755781
	Some victims are forced into street begging.	0.759511
	Children, particularly Romani children, are recruited for forced begging in Poland.	0.763812
	Georgian, Romani, and Kurdish children are subjected to forced begging or coerced into criminality.	0.765216
	Children in street vending or begging are reportedly vulnerable to forced labor.	0.766867
	In Dakar alone, approximately 30,200 talibes beg in the streets.	0.767311
	Street children are sometimes coerced into criminality or forced to beg; begging ringmasters sometimes main children to increase their earnings.	0.776786
	Some street children may be subjected to forced begging or coerced into criminality.	0.777112
	Some street children may be subjected to forced begging or coerced into criminality.	0.777112

Figure 2: Themes Report Derived from SAS Text Miner Text Clustering

Similar results are available across all clusters and are indicative of a variety of themes in human trafficking. This includes where sex trafficking victims are typically exploited, groups who are subject to forced marriage and domestic servitude, what characteristics make individuals most vulnerable to human trafficking, and how debt bondage plays into human trafficking. Similar capabilities are available through the topics capability of both SAS Visual Text Analytics and SAS Contextual Analysis.

IDENTIFYING FOCUSED PATTERNS RELATED TO HUMAN TRAFFICKING

A second method to identify themes in the data is through a term map. In this method of interactive exploration, the user selects a term from the full list of extracted terms across all trafficking reports, and selects to view a term map of related terms and phrases. The user is then presented with a visual depiction of other terms and phrases that tend to be connected to the source term or phrase.

The example below depicts a term map surrounding the term “shelter”. This links other key terms and phrases, such as “provide” and “psychological”, indicating that shelters often provide psychological assistance. Another key linkage includes “female victim”, denoting who the shelters primarily serve. Finally, the term “medical” and “legal” tend to be associated with shelters, indicating other types of aid that are received at shelters. In the example below, in the 80 sentences across all reports that contain the term “shelter”, 44 of them also contain the terms “medical” and “legal”. This type of analysis provides quantitative data to advance anti-trafficking reforms, examine where resources are most needed, and can assist in determining where methods of providing assistance have been proven to be helpful. These methods can subsequently be implemented elsewhere.

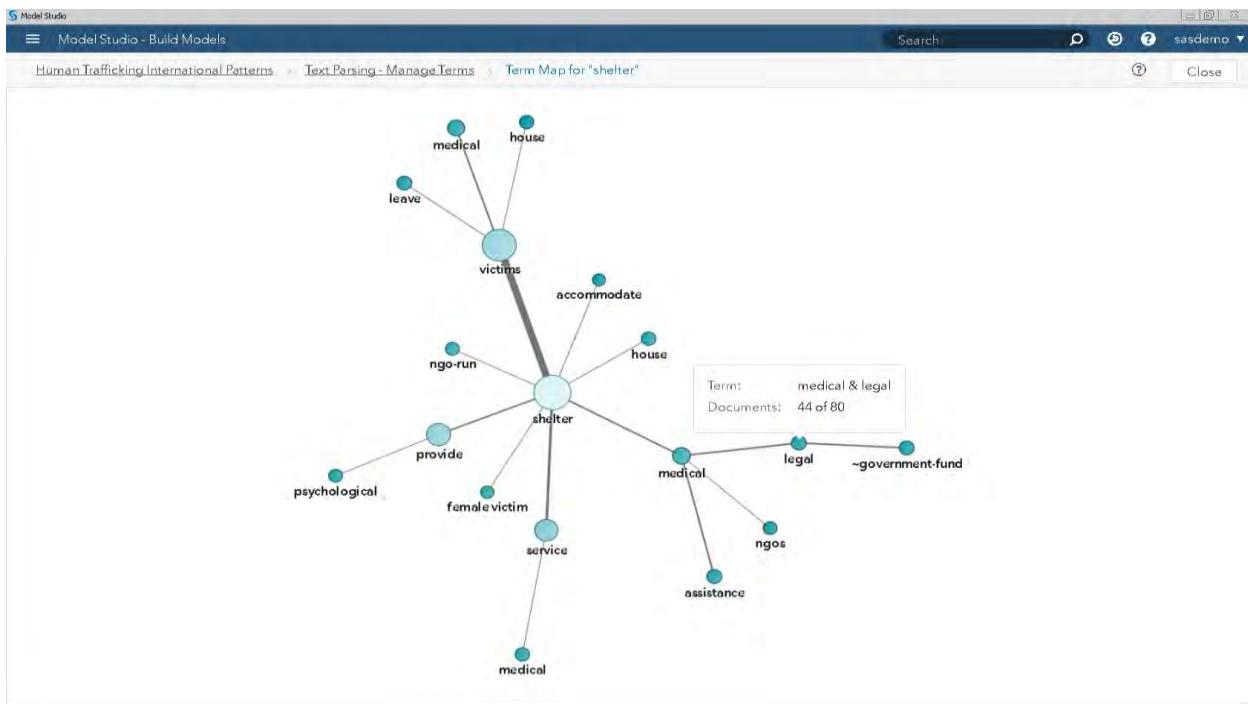


Figure 3: Term Map from SAS Visual Text Analytics Depicting Terms and Phrases Interconnected with the Term "Shelter"

Similar capability is available from the Text Filter node of SAS Text Miner, as well as from the Terms panel of SAS Contextual Analytics.

SAS Visual Text Analytics includes a unique capability across the SAS Text Analytics products that can identify term and phrase similarities to terms of interest. This differs from the term map capabilities in that algorithms identify terms used in a similar context to the selected term. From the Terms node of SAS Visual Text Analytics, the user can select a term, such as "source" in the example below, and view terms and phrases used in a similar context. In this visualization, SAS identifies terms including "source country", "transit country", and "destination country" used in a similar context, indicating that there are connective patterns in the text between countries that are a source of human trafficking victims, and countries where these victims become involved in human trafficking. The visualization also shows these terms in the context of the sentences in which they appear.

The screenshot shows the SAS Model Studio interface with the title "Model Studio - Build Models". Under "Human Trafficking International Patterns", it displays "Term similarities for 'source'" and "Dropped Terms".

Term similarities for "source":

Term	Similarity	Role
source	1.000	N
destination country	0.929	nlpNounGroup
destination	0.901	N
transit	0.895	N
source country	0.851	nlpNounGroup
transit country	0.700	nlpNounGroup
profile	0.593	N
principally	0.577	ADV
past	0.569	N

Dropped Terms:

Term	Role
the	DET
and	CONJ
to	PPOS
of	PPOS
in	PPOS
be	V
for	PPOS
a	DET
or	CONJ
with	PPOS

Documents: 186

All Matched

sentence

...List, Qatar is a **destination country** for men and women subjected to forced labor and, to a much lesser extent, forced prostitution.

... **destination country** for a small number of foreign trafficking victims, including sex trafficking victims from Moldova and Poland, and labor trafficking victims from Bangladesh and Serbia;

...a lesser extent; **destination country** for men, women, and children subjected to forced labor and sex trafficking.

... **destination country** for a limited number of foreign trafficking victims, including sex trafficking victims from Moldova and Poland and labor trafficking victims from Bangladesh, China, the Philippines, and Serbia.

Document 1 of 186

Figure 4: Visual Text Analytics Depiction of Term Similarity to the Term "source"

This connection between source, target, and transit countries is worth further exploration. To verify the depth of this pattern, we turn to the SAS Topics capability. In the example below taken from SAS Contextual Analysis, across 827 sentences, SAS identifies a theme (with no user input) between source countries and target countries. This theme also covers victims including men, women, and children, and the two forms of human trafficking, sex trafficking and labor trafficking.

The screenshot shows the SAS Contextual Analysis interface with the title "SAS® Contextual Analysis".

Topics:

- All Topics (9257)
 - women, labor, children, sex trafficking - destination
 - victim, identify, service, shelter, refer
 - sex trafficking, child, women, girls, traffic
 - complyfully, minimum, standard, minimum, elimination
 - watch, lists, watch, previous, place
 - worker, debt, bondage, debt, bondage, migrant
 - convict, prosecute, offender, trafficker, case
 - plan, implement, written, plan, written, waiver
 - force, labor, labor, force, agriculture, service
 - domestic, sentence, domestic, sentence, girls, prostitution
 - united, east, south, middle, east
 - law, anti-trafficking, human, human trafficking, reinforcement
 - child, children, group, report, armed

Documents: 827 of 9,257

Topic > sextrafficking +sex+women,girls+traffic

Burma is a **source country** for men, women, and children subjected to forced labor, and for women and children subjected to sex trafficking in other countries.

Burma is a **source country** for men, women, and children subjected to forced labor, and for women and children subjected to sex trafficking in other countries.

Uzbekistan (Tier 2) is a **source country** for men, women, and children subjected to forced labor and women and children subjected to sex trafficking.

Tajikistan is a **source country** for men, women, and children subjected to forced labor, and women and children subjected to sex trafficking.

Paraguay is a **source country** for women and children subjected to sex trafficking, and for men, women, and children subjected to forced labor.

Figure 5: SAS Contextual Analysis Depicts a Topic Showing Network Connections in Human Trafficking

EXTRACTING PATTERNS IN HUMAN TRAFFICKING FOR NETWORK VISUALIZATION

Now that we have used exploratory text analytics capabilities to identify a pattern of interest, analysts might be interested in geospatially depicting the interconnection between countries on a world map over time. To prepare for this activity, it is necessary to develop rules to extract these patterns or facts via extraction rules. SAS Visual Text Analytics and SAS Contextual Analysis provide the capability to use a SAS proprietary rule-writing language called LITI to define parameters for fact extraction. Through the SAS Visual Text Analytics interface using LITI, we define rules to extract the victims of trafficking in context, including men, women, and children. This is depicted in the following example visualization of the rule editor and tester below.

The screenshot shows the SAS Model Studio interface with the title "Model Studio - Build Models". The main window is titled "Human Trafficking International Patterns" and "Concepts". On the left, there's a sidebar with "Concepts (14)" and a tree view of "Predefined Concepts (0)" and "Custom Concepts (14)", which includes items like "generalLocation", "getDestinationCountry", and "typeOfPerson". The main area is titled "Edit a Concept" and contains a code editor with the following content:

```
1 CONCEPT_RULE:(ORD, "targetLocation", "natureOfFlow", "country", "_c{typeOfPerson}")
```

Below the code editor, a message says "Code is valid." There are tabs for "Documents (228)" and "Test Sample Text". Under "Test Sample Text", there are four rows of text, each with a "sentence" column and a "Fact Matches" column (both currently show 0). The sentences are:

- ...limited number of **women** and **children** subjected to forced labor and sex trafficking.
- ...source country for **men**, **women**, and **children** subjected to forced labor and for **women** and **children** subjected to sex trafficking.
- ...destination country for **men** and **women** subjected to forced labor and, to a lesser extent, forced prostitution.
- ...destination country for **men**, **women**, and **children** subjected to sex trafficking and forced labor.

At the bottom, it says "Document 1 of 228".

Figure 6: LITI Rules in SAS Visual Text Analytics to Identify Victims of Human Trafficking

These definitions build upon themselves, and some rule definitions, such as a list of country names, become helper definitions when writing rules to extract a larger pattern. A set of rules, along with some post-processing, enables us to extract the full pattern of source countries, destination countries, Boolean indicators indicating the victims of trafficking and types of trafficking, and finally a cooperation indicator derived from the text for each pair of countries to determine whether they are working together to address the trafficking problem. The following screenshot depicts a rule which extracts destination countries for human trafficking victims.

The screenshot shows the SAS Model Studio interface with the 'Build Models' tab selected. In the center, there's a 'Concepts' section where a rule is being edited:

```

1 CONCEPT_RULE:(SENT, "natureOfFlow", "typeofPerson", (OR, "typeofLaborSex",
2 "typeOfLaborLabor"), "_c{generalLocation2}")
3

```

A message below says 'Validation is out of date.' Below this, there are tabs for 'Documents' (240) and 'Test Sample Text'. Under 'Test Sample Text', the 'Matched' tab is selected, showing three sentences from reports:

- ...In Persons Report.....Report.....OMAN: Tier 2 Watch List Oman is a destination and transit country for men and women, p and East Africa, subjected to forced labor and, to a lesser extent, sex trafficking.
- ...2 Watch List.....Tanzania is a source, transit, and destination country for men, women, and children subjected to forced lab...
- ...Persons Report.....Tier 2El Salvador is a source, transit, and destination country for women, men, and children subjecte forced labor.

At the bottom left, it says 'Document 1 of 240'.

Figure 7: Concept Extraction in SAS Visual Text Analytics to Capture Human Trafficking Patterns

The purpose of SAS Visual Text Analytics is to develop and score these rules against source data to generate an additional data set used for visualization and interpretation of the data. In this case, we score the rules developed above to extract source/destination country patterns against the full 63,648 rows of sentence level data. In prior SAS Global Forum submissions, we've explored the output of a text analytics exercise in dashboard format, such as in assessing consumer financial complaints⁵. These past use cases had the benefit of additional structured data in conjunction with the free-text field, such as a user complaint in context of structured geographical information, date of claim, type of claim, and whether a user who submitted the complaint received some form of monetary compensation. In this case, we develop a visualization using only structured data we generated from the unstructured reports, namely, the connection between the countries, including victim information, year of the report, type of trafficking, and cooperation indicator. Consider that we applied automated analysis to turn reams of text into visualization-ready structured data. Consider also that these processes could be immediately deployed for new sets of these reports as they emerge in 2018, 2019 and beyond! A snapshot of this data generated from Visual Text Analytics after postprocessing is depicted below.

ht_tip_connections											
	year	men	women	children	forced_labor	sex_trafficking	base_country	relation_country	cooperation	sentence	
2600	2017	0	1	1	0	1	afghanistan	iran	y	Afghan women and girls are subjected to sex trafficking.	
2601	2017	0	1	1	0	1	afghanistan	pakistan	n	Afghan women and girls are subjected to sex trafficking.	
2602	2017	1	0	1	1	0	afghanistan	turkey	n	Afghan boys and men are subjected to forced labor and...	
2603	2017	1	0	1	1	0	afghanistan	greece	n	Afghan boys and men are subjected to forced labor and...	
2604	2017	1	0	1	1	0	afghanistan	pakistan	n	Afghan boys and men are subjected to forced labor and...	
2605	2017	1	0	1	1	0	afghanistan	iran	y	Afghan boys and men are subjected to forced labor and...	
2606	2017	0	1	1	0	1	afghanistan	china	n	There were reports of women and girls from the Philippin...	
2607	2017	0	1	1	0	1	afghanistan	sri lanka	n	There were reports of women and girls from the Philippin...	
2608	2017	0	1	1	0	1	afghanistan	tajikistan	n	There were reports of women and girls from the Philippin...	
2609	2017	0	1	1	0	1	afghanistan	iran	y	There were reports of women and girls from the Philippin...	
2610	2017	0	1	1	1	1	There were reports of women and girls from the Philippines, Pakistan, Iran, Tajikistan, Sri Lanka, and China subjected to sex trafficking.				
2611	2017	0	1	1	0	1	afghanistan	principles	n	There were reports of women and girls from the Philippin...	
2612	2017	0	0	0	0	1	albania	united kingdom	n	Albanian victims are subjected to sex trafficking in countr...	
2613	2017	0	0	0	0	1	albania	italy	y	Albanian victims are subjected to sex trafficking in countr...	
2614	2017	0	0	0	0	1	albania	greece	n	Albanian victims are subjected to sex trafficking in countr...	
2615	2017	0	0	1	1	0	albania	united kingdom	n	NGOs report an increase in the number of Albanian child...	
2616	2017	0	0	0	1	1	albania	nigeria	n	Foreign victims from European countries, Philippines, an...	

Figure 8: Visualization-Ready Data Generated by Scoring Rules from SAS Visual Text Analytics

NETWORK VISUALIZATION

We load the data generated from the text analytics exercise into SAS Visual Analytics. The following visualizations were accomplished on SAS Viya, but similar visualizations are available on SAS 9. Once the data is loaded and the option to create a new report is selected with that data, we select the Network Analysis object for our geospatial visualization. We select the option under “Network Display” to enable a Map background, which leverages OpenStreetMap by default. We convert the base country and relation country from a categorical data variable to a geography data variable based on the country name. These are set as source and target “roles” for the Network Analysis object. The link width is set to the frequency of connections between source and target countries, enabling thicker lines for relationships that span multiple years. The link color is set to the cooperation_indicator, highlighting links that involve cooperation between source and target countries in orange. Finally, the directionality of the links is assigned under the Link Direction option of the “Network Display” to “Source”, to show the links from source country to destination countries. The resulting diagram, initially centered around South Africa, is shown below.

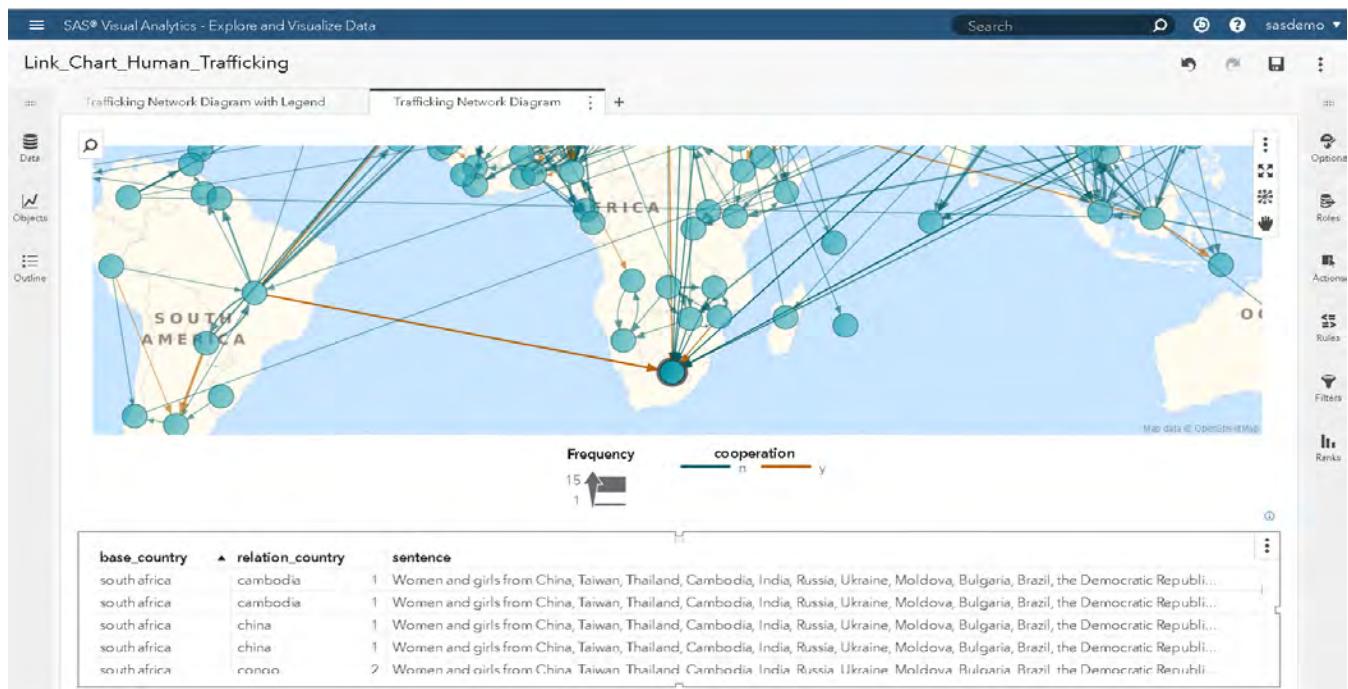


Figure 9: Network Analysis Diagram Showing Patterns of Trafficking in the Southern Hemisphere

This visualization displays the interconnection between all countries across the TIP reports from 2013-2017. It highlights groups of countries involved in trafficking with each other, such as the various countries in the south of Africa as well as South America. It also highlights countries that serve as hubs for larger international trafficking patterns. For each node selected, SAS Visual Analytics displays the text associated with those connections. In this case, it highlights lines from the TIP reports identifying victims of human trafficking in South Africa from source countries including China, Taiwan, Thailand, Cambodia, India, Russia, and Brazil. From here, the text can be assessed to verify the authenticity of the links. Some links, including the link between Brazil and South Africa, are depicted in orange. This shows that SAS identified a relationship in the text between those two countries indicating that they were working together to address the trafficking problem.

Connections between Nigeria and other African countries, as well as to countries in Europe and Asia are particularly strong as shown in the diagram below. This might warrant an analysis of other circumstantial evidence surrounding Nigeria, and we will explore this further in the discussion section of this paper.

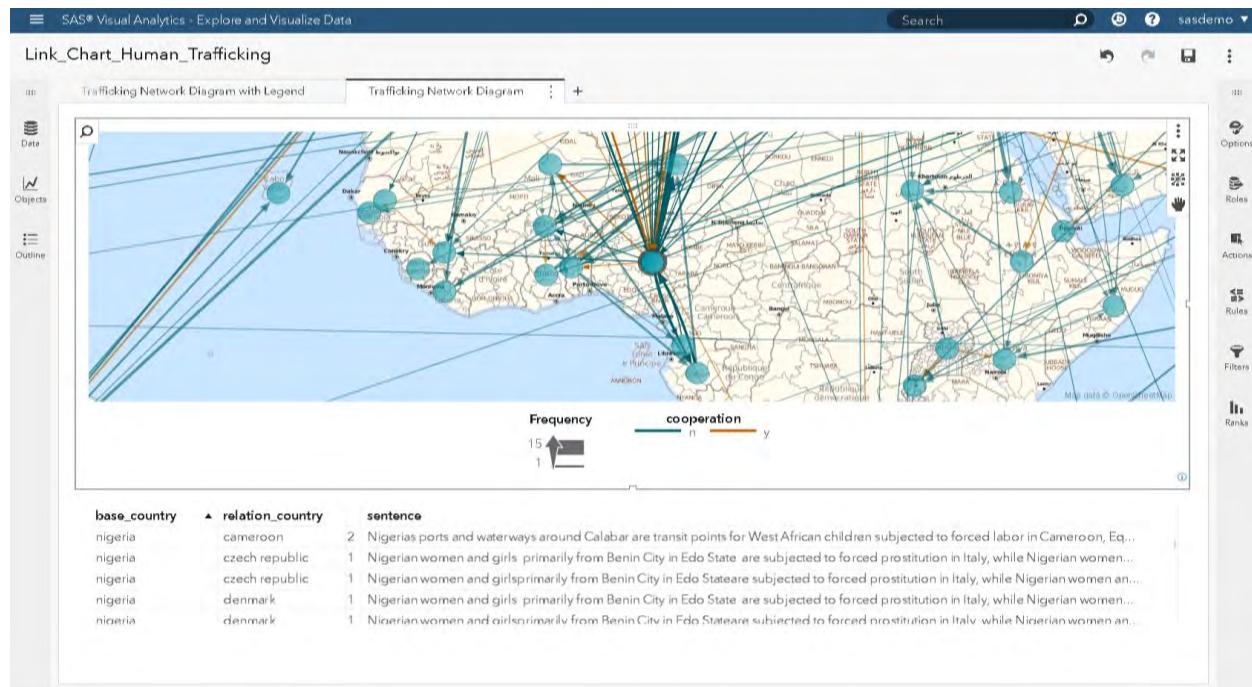


Figure 10: Network Analysis Diagram Highlighting Patterns of Trafficking Surrounding Nigeria

Filters can be applied that showcase certain aspects of trafficking, such as labor trafficking only, or sex trafficking only. In the visualization below, only the patterns of trafficking extracted from the TIP reports that mention children are shown.

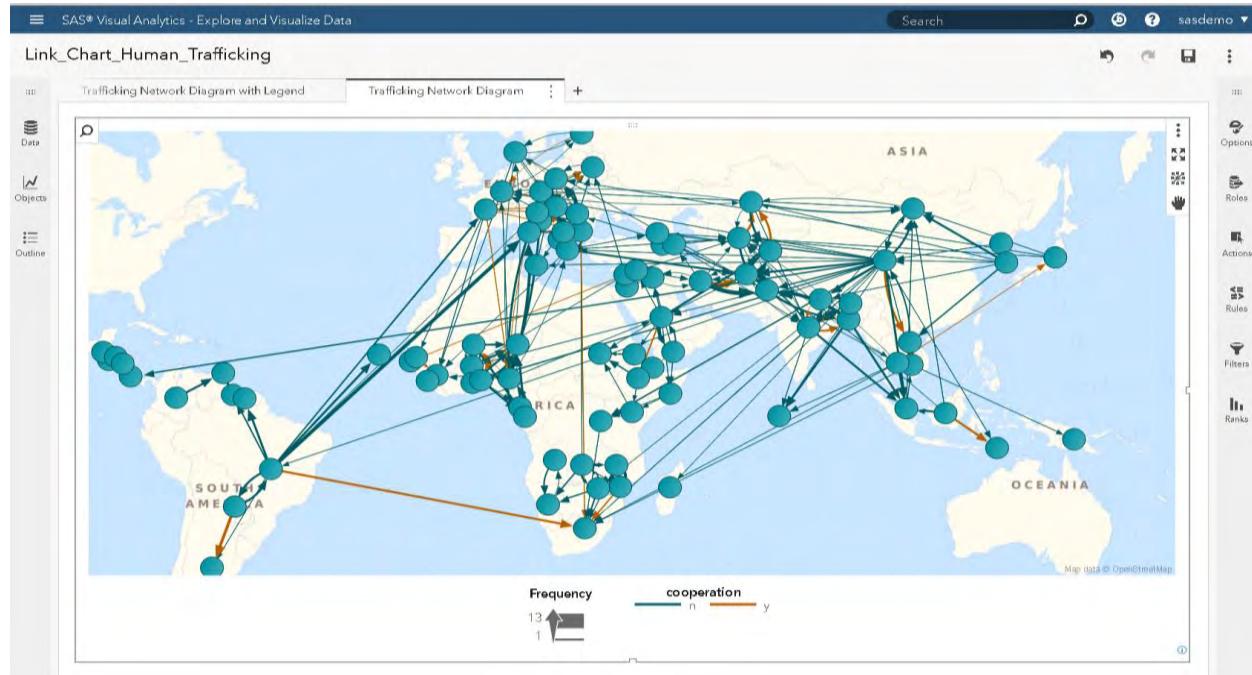


Figure 11: Network Analysis Diagram Depicting International Patterns of Child Trafficking

Finally, in considering visualization and interconnectedness between countries, the single links available in the TIP reports play into a much broader picture. Reports might mention connections such as “Nigeria is a source country for trafficking in other countries including...”. These single node-to-node connections

become much more insightful when seen in the context of all the other node-to-node connections. This is particularly illuminating when countries are specifically called out as transit countries, and these connections in turn reveal second-degree and third-degree connections between source and destination countries. The following visualization reveals Thailand cited as a transit country for a variety of source and destination countries, revealing a larger pattern of international human trafficking.

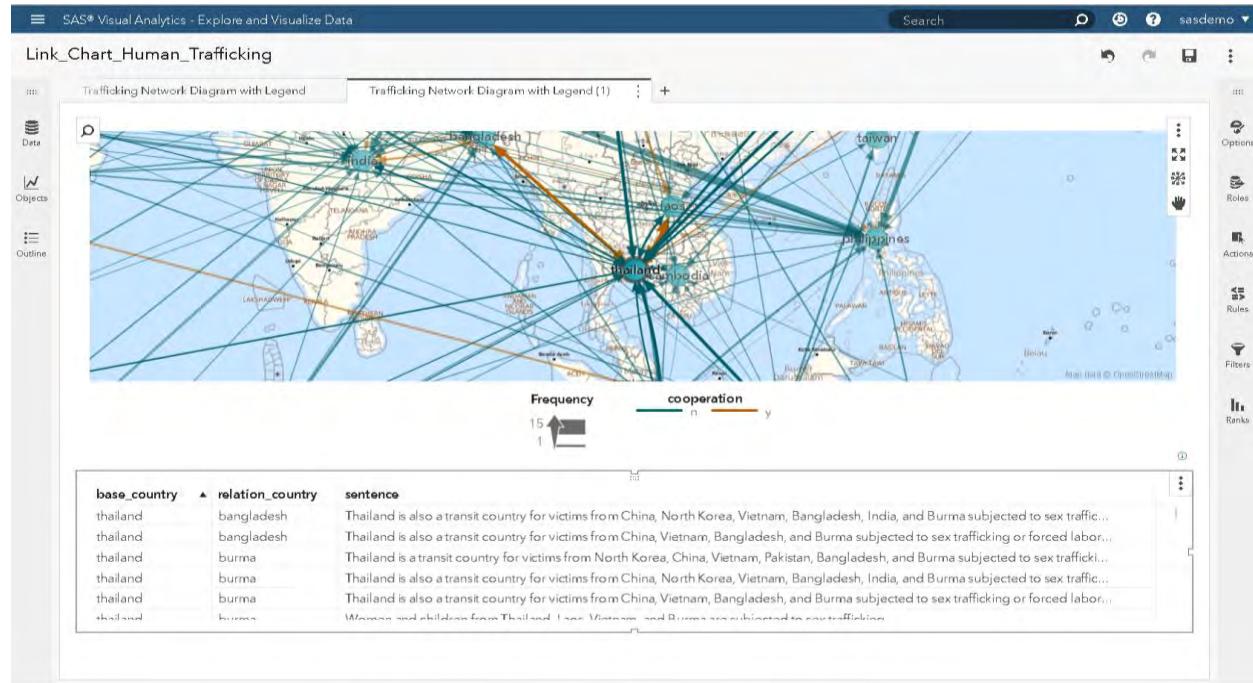


Figure 12: Network Analysis Diagram Depicting Thailand as a Transit Country for International Trafficking

CONCLUSION AND DISCUSSION

In this paper, we showed how SAS could be used to obtain TIP reports from the US Department of State, identify patterns across those reports, and visually depict those patterns. We used the text analysis and visualization capabilities of SAS to answer three questions. First, we identified general trends in the TIP reports. Second, we identified focused themes, including themes around victims seeking shelter internationally. Finally, we extracted a geospatial pattern across all trafficking reports between source and target countries. We then depicted this visually in a network analysis diagram. The network analysis diagram included controls for filtering on trafficking victims, trafficking type, and the year of the report. These results enhance the ability of the U.S. Government and foreign nations worldwide to engage in dialogs advancing anti-trafficking reforms, and to examine where resources are most needed.

The analysis effort to identify the source and destination countries took approximately three days of dedicated effort. Contrast this with a manual effort to extract the same information from the reports. If we approximate 30 minutes per report to identify all the relevant connections that occurred in the data, with approximately 1000 reports, this would require 3 months of effort, or 30 times the time investment. Also, consider that the automated rules can score reports in upcoming years for connections, including 2018, 2019 and beyond at little extra time investment.

Analysis relies on the quality of the underlying data, and all analysis is fraught with challenges involving precision and recall. Precision in this case involves extracting only correct links, including getting the directionality of the connection correct. Recall involves extracting all of the links. Precision, in this analysis, can be improved by developing additional rules to ensure directionality accuracy in the links. Recall in this data set was influenced by factors including generality of information in the TIP reports. For example, the United States does not feature in any of the network links, as the United States is discussed

in general terms in the reports, indicating that the United States is a source country and destination country for trafficking with a variety of foreign nations. This means that SAS is unable to extract a specific pattern related to the United States since specific countries it is connected to are not called out in the reports directly. Such insight provides additional feedback to the analysts developing these reports in terms of where specific patterns need to be built out upon the more general patterns. Such work can enhance understanding of the international patterns between several degrees of source and destination countries.

There are several different trafficking-related use cases, including drug trafficking and weapons trafficking. These tend to tie together with the human trafficking element. Other organizations who could potentially benefit from a trafficking solution include federal, state, and local law enforcement agencies. A solution that assesses and prioritizes trafficking-related leads could be set up from a law enforcement perspective, but could also address victim assistance. Regarding data that would assist law enforcement, search engines for classified ads become a repository of data that plays into human trafficking, particularly sex trafficking. They can be assessed to identify geographically where recruitment ads are spiking, where there are similar or emerging patterns in ads, and can ultimately assist law enforcement in identifying networks of trafficking-based organizations. There is a trafficking related pattern to data from financial organizations as well, including the major banks. The Financial Crimes Enforcement Network (FinCEN) has published guidelines to banks on recognizing activity that might be associated with human smuggling and human trafficking⁶.

As mentioned, there are different sources of data that support the use case to assess patterns of international human trafficking. For example, to identify why Nigeria has a number of trafficking connections to a variety of countries in Africa, Europe, and Asia, we can examine data sources such as the Armed Conflict Location and Event Data project⁷ (ACLED) to look for connections. In addition, we can apply machine learning and auto-categorization to the ACLED data as prescribed in a previous SAS Global Forum paper published in 2016⁸.

In the screenshot below, we used a categorical hierarchy developed with machine learning against the ACLED data to explore themes in violence against civilians in Nigeria and the surrounding regions. The visualization depicts specific recorded instances of abduction and kidnaping, and drills down to the event text describing what happened. There is significant event traffic in Nigeria, depicting a destabilizing force that contributes to the vulnerability of its citizens to human trafficking. This analysis adds to the current evidence that many Nigerians seek work abroad due to extreme poverty, and are subsequently exploited for forced labor and prostitution. The data available from the TIP reports and the ACLED project is further reinforced by an exposé by CNN, where individuals who have sought work abroad as migrants from Niger and Nigeria among other African countries are sold at a slave auction⁹.

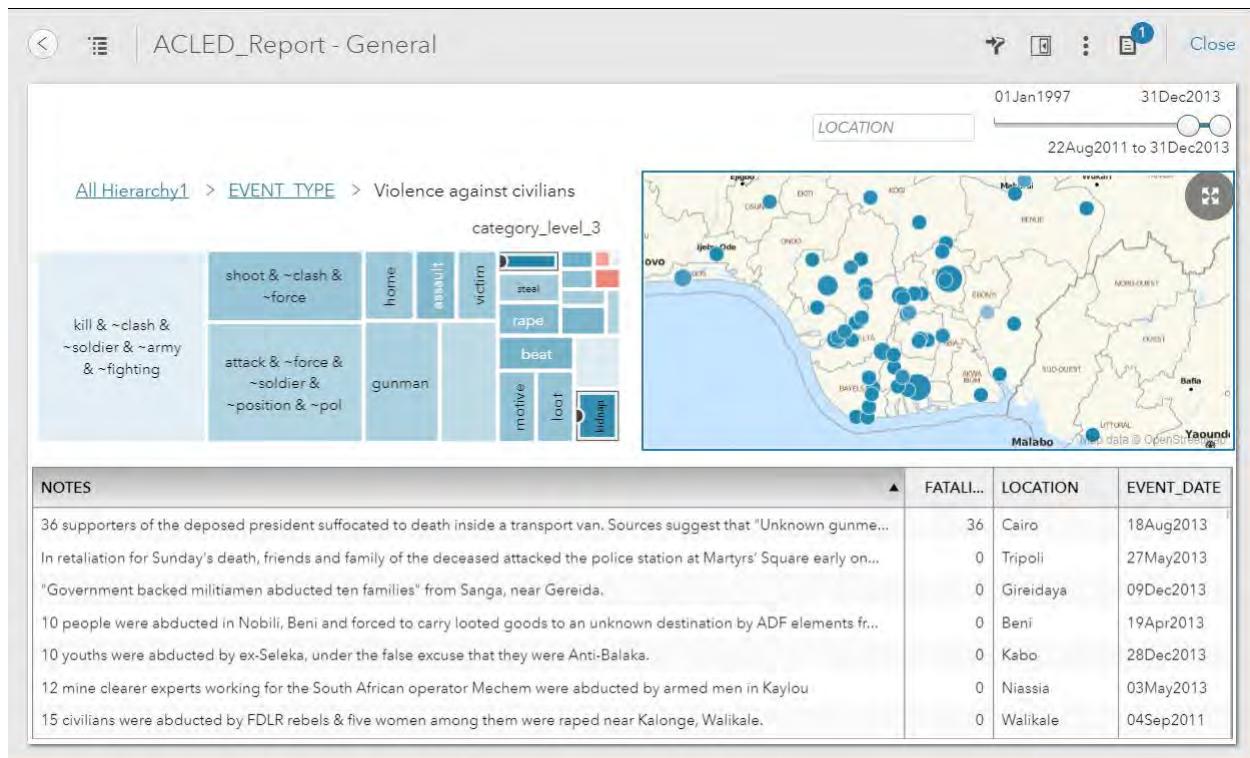


Figure 13: Visualization Depicting Kidnapping and Abduction Events in Nigeria and the Surrounding Countries Using Data from the ACLED Project

In summary, the analytics and visualizations presented here are an effort to show how data related to human trafficking can be transformed into actionable information. By taking advantage of data and analytics, data scientists and researchers are able to shine light on the problem, and thereby help international government, law enforcement, and victims advocacy groups find better ways to address it¹⁰.

REFERENCES

1. U.S. Department of State. 2017. "Trafficking in Persons Report." Accessed February 2, 2018. <https://www.state.gov/j/tip/rls/tiprpt/>.
2. The Polaris Project. 2018. "The Facts." Accessed February 2, 2018. <https://polarisproject.org/human-trafficking/facts>.
3. Figallo-Monge, Manuel. "Pedal-to-the-Metal Analytics with SAS® Studio, SAS® Visual Analytics, SAS® Visual Statistics, and SAS® Contextual Analysis" Proceedings of the SAS Global Forum 2016 Conference. Cary NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings16/SAS6560-2016.pdf>.
4. Albright, Russ. Cox, James. Jin, Ning. 2016. "Getting More from the Singular Value Decomposition (SVD): Enhance Your Models with Document, Sentence, and Term Representations" Proceedings of the SAS Global Forum 2016 Conference. Cary NC: SAS Institute Inc. Available <https://support.sas.com/resources/papers/proceedings16/SAS6241-2016.pdf>.
5. Sabo, Tom. 2017. "Applying Text Analytics and Machine Learning to Assess Consumer Financial Complaints." *Proceedings of the SAS Global Forum 2017 Conference*. Cary NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings17/SAS0282-2017.pdf>.

6. United States Department of the Treasury, Financial Crimes Enforcement Network. 2014. "Advisory Information". Accessed February 8, 2018.
<https://www.fincen.gov/resources/advisories/fincen-advisory-fin-2014-a008>.
7. ACLED Data; Bringing Clarity to Crisis. 2018. "About". Accessed February 12, 2018.
<http://www.acleddata.com/>.
8. Sabo, Tom. 2016. "Extending the Armed Conflict Location and Event Data Project with SAS® Text Analytics." Proceedings of the SAS Global Forum 2016 Conference. Cary NC: SAS Institute Inc. Available <https://support.sas.com/resources/papers/proceedings16/SAS6380-2016.pdf>.
9. CNN. 2017. "Migrants being sold as slaves." Accessed February 12, 2018.
<http://www.cnn.com/videos/world/2017/11/13/libya-migrant-slave-auction-lon-orig-md-ejk.cnn>.
10. SAS. 2017. "Analytics tackles the scourge of human trafficking." Accessed February 12, 2018.
https://www.sas.com/en_us/insights/articles/analytics-analytics-tackles-human-trafficking.html.

ACKNOWLEDGMENTS

Thanks to Emily McRae and John Dillman for assisting with the visualizations, and thanks also to Emily for her review of this paper. Thanks to Mary Beth Ainsworth for her insight into the human trafficking problem nationally and internationally. Also, thanks to Dr. James R. Van Scotter for providing awareness into how local and state law enforcement can use data to identify human trafficking trends within the U.S.

RECOMMENDED READING

- Sabo, Tom. 2014. SAS Institute white paper. "*Text Analytics in Government: Using Automated Analysis to Unlock the Hidden Secrets of Unstructured Data.*" Available
http://www.sas.com/en_us/whitepapers/text-analytics-in-government-106931.html.
- Chakraborty, G., M. Pagolu, S. Garla. 2013. *Text Mining and Analysis; Practical Methods, Examples, and Case Studies Using SAS®*. SAS Institute Inc.

- Sabo, Tom. 2014. “*Uncovering Trends in Research Using Text Analytics with Examples from Nanotechnology and Aerospace Engineering.*” *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings14/SAS061-2014.pdf>
- Sabo, Tom. 2015. “*Show Me the Money! Text Analytics for Decision-Making in Government Spending.*” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings15/SAS1661-2015.pdf>.
- Reamy, Tom. 2016. Deep Text; Using Text Analytics to Conquer Information Overload, Get Real Value from Social Media, and Add Big(ger) Text to Big Data. Medford NJ: Information Today, Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Tom Sabo, Principal Solutions Architect
 1530 Wilson Blvd.
 Arlington, VA 22209
 SAS Federal LLC
 +1 (703) 310-5717
tom.sabo@sas.com
 @mrTomSab

Adam Pilz, Senior Solutions Architect
 121 W Trade St.
 Charlotte, NC 28202
 SAS Institute Inc
 +1 (919) 348-6039
adam.pilz@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Paper SAS1961-2018

Biomedical Image Analytics Using SAS® Viya®

Fijoy Vadakkumpadan and Saratendu Sethi, SAS Institute Inc.

ABSTRACT

Biomedical imaging has become the largest driver of health care data growth, generating millions of terabytes of data annually in the US alone. With the release of SAS® Viya™ 3.3, SAS has, for the first time, extended its powerful analytics environment to the processing and interpretation of biomedical image data. This new extension, available in SAS® Visual Data Mining and Machine Learning, enables customers to load, visualize, process, and save health care image data and associated metadata at scale. In particular, it accommodates both 2-D and 3-D images and recognizes all commonly used medical image formats, including the widely used Digital Imaging and Communications in Medicine (DICOM) standard. The visualization functionality enables users to examine underlying anatomical structures in medical images via exquisite 3-D renderings. The new feature set, when combined with other data analytic capabilities available in SAS Viya, empowers customers to assemble end-to-end solutions to significant, image-based health care problems. This paper demonstrates the new capabilities with an example problem: diagnostic classification of malignant and benign lung nodules that is based on raw computed tomography (CT) images and radiologist annotation of nodule locations.

INTRODUCTION

Biomedical Image processing is an interdisciplinary field that is at the intersection of computer science, machine learning, image processing, medicine, and other fields. The origins of biomedical image processing can be attributed to the accidental discovery of X-rays by Wilhelm Conrad Roentgen in 1895. The discovery made it possible for the first time in the history of humans to noninvasively explore inside the human body before engaging in complex medical procedures. Since then, more methods for medical imaging have been developed, such as computed tomography (CT), magnetic resonance imaging (MRI), ultrasound imaging, single-photon emission computed tomography (SPECT), positron emission tomography (PET), and visible-light imaging. The goal of biomedical image processing is to develop computational and mathematical methods for analyzing such medical images for research and clinical care. The methods of biomedical image processing can be grouped into following broad categories: image segmentation (methods to differentiate between biologically relevant structures such as tissues, organs, and pathologies), image registration (aligning images), and image-based physiological modeling (quantitative assessment of anatomical, physical, and physiological processes).

SAS has a rich history of supporting health and life sciences customers for their clinical data management, analytics, and compliance needs. SAS® Analytics provides an integrated environment for collection, classification, analysis, and interpretation of data to reveal patterns, anomalies, and key variables and relationships, leading ultimately to new insights for guided decision making. Application of SAS® algorithms have enabled patients to transform themselves from being passive recipients to becoming active participants in their own personalized health care. With the release of SAS Viya 3.3, SAS customers can now extend the analytics framework to take advantage of medical images along with statistical, visualization, data mining, text analytics, and optimization techniques for better clinical diagnosis.

Images are supported as a standard SAS data type in SAS Visual Data Mining and Machine Learning, which offers an end-to-end visual environment for machine learning and deep learning—from data access and data wrangling to sophisticated model building and deployment in a scalable distributed framework. It provides a comprehensive suite of programmatic actions to load, visualize, process, and save health care image data and associated metadata at scale in formats such as Digital Imaging and Communication in Medicine (DICOM), Neuroimaging Informatics Technology Initiative (NIFTI), nearly raw raster data (NRRD), and so on. This paper provides a comprehensive overview of the biomedical image processing capabilities in SAS Visual Data Mining and Machine Learning by working through real-world scenarios of building an end-to-end analytic pipeline to classify malignant lung nodules in CT images.

END-TO-END BIOMEDICAL IMAGE ANALYTICS IN SAS VIYA

SAS® Viya™ uses an analytic engine known as SAS® Cloud Analytic Services (CAS) to perform various tasks, including biomedical image analytics. Building end-to-end solutions in SAS Viya typically involves assembling CAS actions, which are the smallest units of data processing that are initiated by a CAS client on a CAS server. CAS actions are packaged into logical groups called action sets. Presently, two action sets, *image* and *bioMedImage*, host actions that directly operate on biomedical imagery.

The *image* action set contains two actions for biomedical image analytics: the *loadimages* action loads biomedical images from disk into memory, and the *saveimages* action saves the loaded images from memory to disk. These actions support all common biomedical image formats, including the DICOM standard, which is widely used in clinical settings. The *bioMedImage* action set currently includes three actions, *processBioMedImages*, *segmentBioMedImages*, and *buildSurface*, for preprocessing, segmentation, and visualization of biomedical images, respectively. At this time, full support is available only for two- and three-dimensional (2-D and 3-D), single-channel biomedical images in these action sets.

The output produced by the actions in the *image* and *bioMedImage* action sets can be used as input to other actions, such as those in action sets for machine learning (ML) and artificial intelligence (AI), to derive insights that inform decisions. Figure 1 presents an end-to-end biomedical image analytics pipeline in SAS® Viya™. On one end of the pipeline are raw image data and metadata on disk, and on the other end are helpful insights that can inform decisions. The major steps in the pipeline, along with the primary action sets (in italics) that can be used to implement those steps, are displayed in rectangular boxes. Examples of ML and AI action sets include the *pca* action set, which performs principal component analysis (PCA), and the *deepLearn* action set, which performs deep learning.

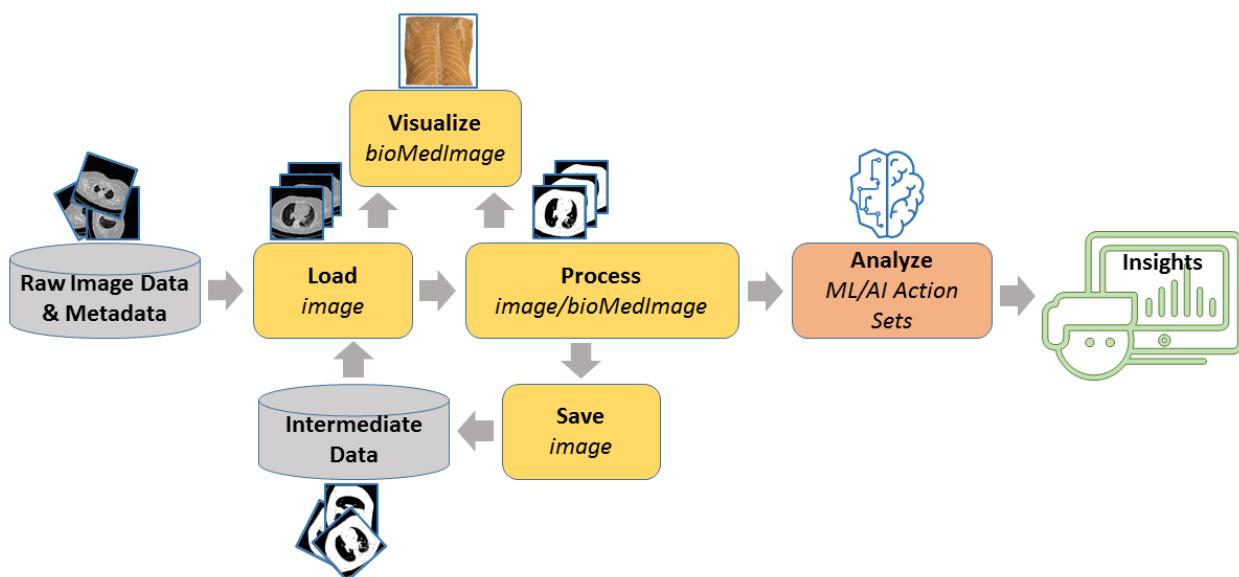


Figure 1. Processing Pipeline for End-to-End Biomedical Image Analytics in SAS Viya

LUNG NODULE CLASSIFICATION: AN EXAMPLE USE CASE

This section illustrates the pipeline shown in Figure 1 by demonstrating how to build an end-to-end solution that can assist with a real-world biomedical image analytics problem, specifically lung nodule classification that is based on 3-D CT images of patient torsos and radiologist annotations of nodule locations. Lung nodules are lumps of dead tissue that commonly occur in humans, less than 5% of which are malignant (McWilliams et al. 2013). Radiologists are responsible for determining whether a nodule visually observed in a patient image is potentially cancerous so that a definitive test such as biopsy is performed for that patient. This paper focuses on nodule shape, one of many factors that radiologists account for in their classification (Niehaus, Raicu, Furst, and Armato 2015). The basis for a shape-based classification is the irregular protrusions (called spiculations) that commonly exist on the surfaces of

malignant nodules. Benign nodules, on the other hand, have smooth and spherical surfaces more often than not (Niehaus, Raicu, Furst, and Armato 2015). This example demonstrates two solutions that can assist with the classification, one based on ML and the other on AI. All client-side source code in this demonstration was written in Python. The SAS Scripting Wrapper for Analytics Transfer (SWAT) package was used to interface with the CAS server, and the Mayavi library (Ramachandran and Varoquaux 2011) was used to perform 3-D visualizations of image-based data.

DATA SELECTION AND PREPROCESSING

All patient data used in this paper were downloaded from The Cancer Imaging Archive (TCIA) (Armato et al. 2015; Armato et al. 2016; Clark et al. 2013). The TCIA data consist of 3-D, thoracic, transaxial, CT images of patients in DICOM format (Figure 2A), radiologist annotations of centers of one or more lung nodules per image, and the definitive diagnoses of each nodule as benign or malignant. The in-plane pixel size of the images ranged from 0.549 to 0.900 mm, and the slice thickness was 1mm. Since the goal was to demonstrate the capabilities of the SAS Viya, and not to invent a clinically significant method for lung nodule classification, only a small set of 10 nodules (5 benign and 5 malignant) from the TCIA data set was included in the analyses. For each of these nodules, a 2-D bounding box around the nodule in the slice that contains the radiologist-annotated nodule center was manually identified. The final annotation data for each nodule consisted of the patient identifier (PID), index of the slice containing the nodule center, 2-D pixel coordinates of the top left corner of the bounding box, width and height of the bounding box in terms of number of pixels, and definitive diagnosis (Figure 2A). All annotation data were stored in a comma-separated values (CSV) file.

To preprocess the images, all 3-D images were recursively loaded on the server as illustrated by this code snippet:

```
s.image.loadImages(path = '/.../TCIASubset/',
                    casOut = vl(name='origMedical', replace='TRUE'),
                    addColumns = {"POSITION", "ORIENTATION", "SPACING"},
                    recurse = True,
                    series = vl(dicom=True),
                    labelLevels = 1,
                    decode = True)
```

Here, `s` is the session returned by SWAT, and the images were loaded into a CAS table named `origMedical`. Note that the `series` parameter list with `dicom=True` directed the `loadImages` action to assemble 3-D images from the DICOM files. All DICOM files for a patient were stored in a subdirectory of `TCIASubset`, whose name matched the PID of that patient. This, in combination with the `labelLevels` parameter set to 1, meant that the output table had a column named `_label_`, which contained the PID for each image. Next, the annotation data were loaded as follows:

```
s.table.loadTable(path = '/.../TCIAannotations.csv',
                  importoptions = vl(filetype="csv", getNames=True),
                  casout = vl(name='trainlabels', replace=True))
```

Table 1 presents all the data in the CAS table `trainlabels`, which was created by the preceding code. The PIDs in this table are same as the ones in TCIA repository.

Next, from each patient image, a 3-D patch that contained a center portion of the nodule was extracted using the `processBioMedImages` action and saved on disk by using the `saveImages` action. The final preprocessing step was to load all patches into a single CAS table by using the `loadImages` action (Figure 2B). The extraction and saving of the 3-D patches is illustrated in this code snippet:

```
for psn in range(numberOfPatients):
    wclause = "_label_='"+PID[psn]+"'"
    s.bioMedImage.processBioMedImages(
        images = vl(table=vl(name='origMedical', where=wclause)),
        steps = [
            vl(stepParameters=vl(
```

```

        stepType='CROP',
        cropParameters=vl(cropType='BASIC',
                           imageSize=[W[psn],H[psn],2*deltaZ+1],
                           pixelIndex=[X[psn],Y[psn],Slice[psn]-deltaZ])),
        decode = True,
        copyVars = {"_label_","_path_","_type_"},
        addColumns={"POSITION", "ORIENTATION", "SPACING"},
        casOut = vl(name='noduleRegion', replace=True))

s.image.saveImages(
    images = vl(table='noduleRegion', path='_path_'),
    subdirectory = 'TrainDataNoduleRegions/',
    type = 'nii',
    labelLevels = 1)

s.image.loadImages(
    casout = vl(name='nodules3D', replace=True),
    path = '/.../TrainDataNoduleRegions/',
    recurse = True,
    addColumns = {"POSITION","ORIENTATION","SPACING"},
    labelLevels = 1,
    decode = True)

```

The `deltaZ` in the preceding code requests that five slices on either side of a nodule center be selected in creating the 3-D patch for that nodule. Therefore, there were 11 slices in each 3-D patch. The vectors `PID`, `X`, `Y`, `Slice`, `W`, and `H` in the preceding code were created by fetching the annotation table (Table 1) to the client side and extracting its columns. The `labelLevels` parameter in the `loadImages` and `saveImages` action calls ensured that the final table `nodules3D` contained PIDs.

PID	X	Y	Slice	W	H	Diagnosis
CT-Training-LC009	129	279	63	39	43	malignant
CT-Training-BE007	371	190	194	29	32	benign
CT-Training-LC002	132	352	70	14	14	malignant
CT-Training-BE001	396	288	169	12	12	benign
CT-Training-LC003	365	314	70	19	19	malignant
LUNGx-CT002	311	328	205	37	37	benign
LUNGx-CT003	359	359	146	31	31	malignant
LUNGx-CT009	165	200	164	19	19	benign
LUNGx-CT019	114	345	131	36	36	malignant
LUNGx-CT024	97	274	197	20	20	benign

Table 1. Nodule Annotations Loaded from the CSV File

The following annotations are contained in the annotation table:

- PID is the patient identifier
- X and Y are the 2-D coordinates of the nodule bounding box that was drawn
- Slice is the index of the slice that contains the nodule center as determined by a radiologist
- W and H are the width and height of the bounding box
- Diagnosis is the definitive diagnosis for the nodule

The following steps are used in the ML-based solution for lung nodule classification and are illustrated in Figure 2:

- A. Start with raw 3-D image data and annotations (red).
- B. Extract 3-D regions around malignant (top) and benign (bottom) nodule centers from the raw images and annotations.
- C. Segment the nodule regions (left) and visualize the surface (right).
- D. Resample the 2-D slices that were extracted from the segmentations.
- E. View the 2-D slices after morphological operations.
- F. View the histogram of a metric that can discriminate between benign and malignant nodules.
- G. Perform ROC (receiver operating curve) analysis to determine the optimal metric threshold that can help classify a new nodule.

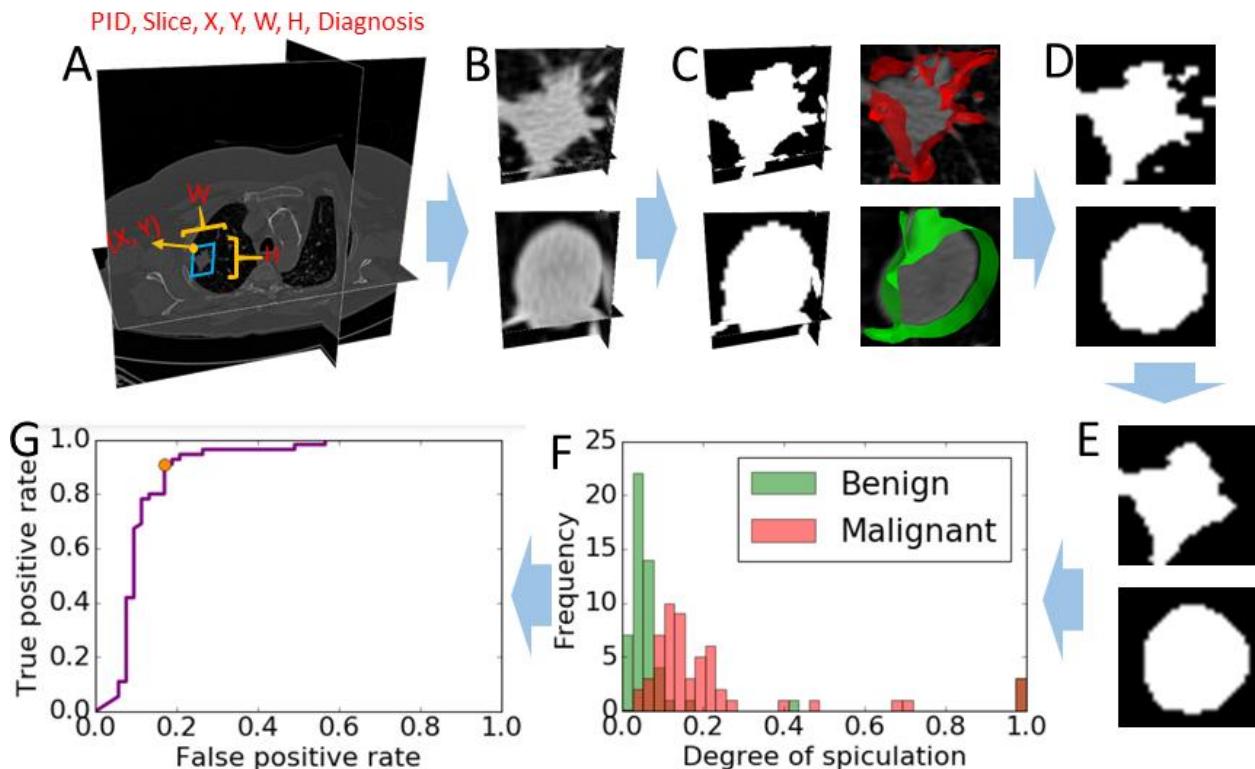


Figure 2. Processing Steps in the ML-Based Solution for Lung Nodule Classification

SOLUTION USING MACHINE LEARNING WITH AN ENGINEERED SHAPE FEATURE

To compute a shape feature, first the 3-D nodule patches were segmented by anisotropic diffusion smoothing followed by Otsu thresholding (Johnson, McCormick, and Ibanez 2017) using the `processBioMedImages` action:

```
s.bioMedImage.processBioMedImages(
    images = vl(table=vl(name='nodules3D')),
    steps = [
        vl(stepParameters=vl(
            stepType = 'SMOOTH',
            smoothParameters = vl(
                smoothType='GRADIENT', iterations=3,
```

```

        timeStep=0.03) )),
vl(stepParameters=vl(
    stepType='THRESHOLD',
    thresholdParameters=vl(
        thresholdType='OTSU',
        regions=2))],
decode = True,
copyVars = {"_label_","_path_","_id_"},
addColumn = {"POSITION", "ORIENTATION", "SPACING"},
casOut = vl(name='masks3D', replace=True))

```

See Figure 2C for example images after segmentation. Note that multiple processing steps are performed in sequence in a single call to the *processBioMedImages* action. Smoothed surfaces of the nodule regions were then constructed using the *buildSurface* action, as follows:

```

s.biomedimage.buildsurface(
    images = vl(table=vl(name='masks3D')),
    intensities = {1},
    smoothing = vl(iterations=3),
    outputVertices = vl(name='noduleVertices', replace=True),
    outputFaces = vl(name='noduleFaces', replace=True))

```

The action produces two output CAS tables, *outputVertices* and *outputFaces*, which contain lists of vertices and triangles of the generated surfaces (one surface per nodule). Surfaces and original gray-scale images that correspond to a few sample nodules were then fetched to the client side and visualized together by using the Mayavi method (Figure 2C), to qualitatively assess the segmentation accuracy.

Next, each segmented 3-D nodule image was split into individual 2-D slices in the transaxial direction so that each slice could be analyzed as a separate observation. This conversion was done using the *EXPORT_PHOTO* feature of the *processBioMedImages* action as follows:

```

s.biomedimage.processBioMedImages(
    images = vl(table=vl(name='masks3D')),
    steps = [vl(stepParameters=vl(stepType='EXPORT_PHOTO'))],
    decode = True,
    copyVars={"_label_"},
    casOut = vl(name='masks', replace=True))

```

The resulting images (Figure 2D) in the *masks* CAS table were in a format that was accepted by the photographic image processing actions in the *image* action set. Then, the *processImages* action was used to resize the 2-D images to have a uniform size of 32x32 and to perform morphological opening (Johnson, McCormick, and Ibanez 2017):

```

pgm = "length _path_ varchar(*);
       _path_=PUT(_bioMedId_*1000+_sliceIndex_, 5.);"
s.image.processImages(
    imageTable = vl(
        name='masks',
        computedVars={"_path_"},
        computedVarsProgram=pgm),
    casOut = vl(name='masksScaled', replace='TRUE'),
    imageFunctions = [
        vl(functionOptions=vl(
            functionType="RESIZE",
            width=32,
            height=32)),
        vl(functionOptions=vl(
            functionType="THRESHOLD",

```

```

        type="BINARY",
        value=0) ) ,
decode=True)

s.image.processImages(
    imageTable = v1(
        name='masks',
        computedVars={"_path_"},
        computedVarsProgram=pgm),
    casOut = v1(name='masksScaled', replace='TRUE'),
    imageFunctions = [
        v1(functionOptions=v1(
            functionType="MORPHOLOGY",
            method="ERODE",
            kernelWidth=3,
            KernelHeight=3)),
        v1(functionOptions=v1(
            functionType="MORPHOLOGY",
            method="ERODE",
            kernelWidth=3,
            KernelHeight=3)),
        v1(functionOptions=v1(
            functionType="MORPHOLOGY",
            method="DILATE",
            kernelWidth=3,
            KernelHeight=3)),
        v1(functionOptions=v1(
            functionType="MORPHOLOGY",
            method="DILATE",
            kernelWidth=3,
            KernelHeight=3))],
decode = True)

```

The preceding code computes a new column, `_path_`, which is used later to uniquely identify each 2-D slice. The thresholding step was necessary after resizing because resizing performs interpolation, which made the image nonbinary. The critical operation here, the morphological opening, performed by the second action call eliminated small and thin regions that constituted a significant part of spiculations. As such, the malignant nodule patches “lost” a significant number of foreground pixels, whereas benign ones retained most of their pixels (Figure 2D and 2E). Based on this result, the shape metric was defined as the relative difference in the number of foreground pixels of a nodule patch between the `masksScaled` and `masksFinal` tables. In the following, this metric is called the degree of speculation (DOS).

To compute the DOS metric for each nodule patch, the `flattenImages` action was used to separate the value of each pixel of that nodule in the `masksScaled` table into individual columns, and the sum of these values was fetched to the client side, as follows:

```

s.image.processImages(
    imageTable = 'masksScaled',
    casOut = v1(name='masksScaledColor', replace='TRUE'),
    imageFunctions = [
        v1(functionOptions=v1(
            functionType="CONVERT_COLOR",
            type="GRAY2COLOR"))],
decode=True)

s.image.flattenImageTable(
    imageTable = 'masksScaledColor',

```

```

casOut = vl(name='masksScaledFlat', replace='TRUE'),
width = 32,
height = 32)

pgm = "nz = c1";
for num in range(2, commonW*commonH*3 + 1):
    pgm += "+c"+str(num)
scaledSum = s.fetch(
    table = vl(
        name='masksScaledFlat',
        computedVars={"nz"},
        computedVarsProgram=pgm),
    fetchVars={'_path_', '_label_', 'nz'},
    to = 1000) ['Fetch']

```

Here, the conversion of the gray-scale images into color was necessary because the *flattenImages* action assumes that all images have three channels. Next, the same sequence of operations was performed on the `masksFinal` table to fetch the sums into another table, `scaledFinal`. Then, the two tables were joined on the `_path_` variable, and the relative difference between the sums in each row was calculated. The histogram of the DOS metric (Figure 2F) shows a bimodal distribution, demonstrating that the metric can discriminate between benign and malignant nodules. A receiver operating characteristic (ROC) analysis revealed an optimal threshold of 0.08 for the metric (Figure 2G). The classification accuracy of the metric was 85% as per a 10-fold cross validation.

SOLUTION USING ARTIFICIAL INTELLIGENCE WITH A CONVOLUTIONAL NEURAL NETWORK

This section uses an alternative solution to assist with lung nodule classification. It uses a convolutional neural network (CNN) to demonstrate the application of artificial intelligence (AI) features that are available in SAS Viya for biomedical image analytics. CNN is a deep learning architecture that has been found to be most effective in image processing. . The following code defines a network, called MicroNet, which has just three main layers (Figure 3), including two convolution + maxpool layers, and one fully connected layer:

```

s.deepLearn.buildModel(
    model = vl(name='microNet', replace=True),
    type='CNN')

s.deepLearn.addLayer(          # Input
    model = 'microNet',
    name = 'images',
    layer = dict(type='input', nchannels=1, width=32, height=32))

s.deepLearn.addLayer(          # First convolution+maxpool layer
    model = 'microNet',
    name = 'conv1',
    layer = dict(type='convolution', nFilters=1, width=3, height=3,
                 stride=1, init='NORMAL', std=0.1, truncationfactor=2,
                 act='RELU'),
    srcLayers = ['images'])

s.deepLearn.addLayer(
    model = 'microNet',
    name = 'pool1',
    layer = dict(type='pooling', width=3, height=3, stride=3, pool='max'),
    srcLayers = ['conv1'])

s.deepLearn.addLayer(          # Second convolution+maxpool layer

```

```

model = 'microNet',
name = 'conv2',
layer = dict(type='convolution', nFilters=2, width=3, height=3,
             stride=1, init='NORMAL', std=0.1, truncationfactor=2,
             act='RELU'),
srcLayers = ['pool1'])
s.deepLearn.addLayer(
    model = 'microNet',
    name = 'pool2',
    layer = dict(type='pooling', width=2, height=2, stride=2, pool='max'),
    srcLayers = ['conv2'])

s.deepLearn.addLayer(          # Fully-connected layer
    model = 'microNet',
    name = 'fc1',
    layer = dict(type='fullconnect', n=2, act='relu', init='NORMAL',
                 std=0.1, truncationfactor=2),
    srcLayers = ['pool2'])

s.deepLearn.addLayer(
    model = 'microNet',
    name = 'outlayer',
    layer = dict(type='output', act='softmax'),
    srcLayers = ['fc1'])

```

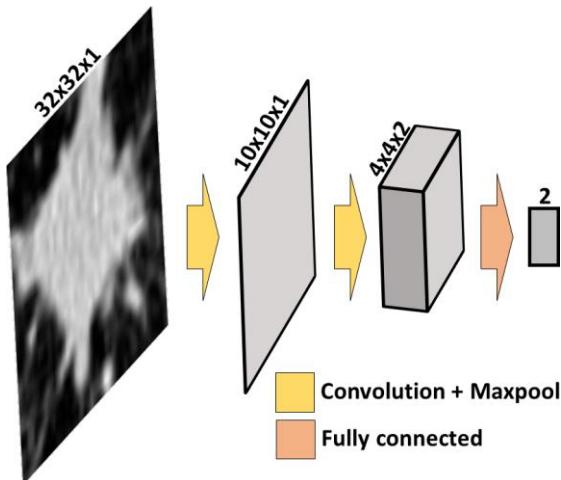


Figure 3. CNN Architecture Used in This Example

The input to the network were 32×32 , 2-D, gray-scale patches that were created from the 3-D nodule regions (see Figure 2B) by using the `RESIZE` and `EXPORT_PHOTO` features of the `processBioMedImages` action. All kernels, except the one used in the maxpool operation in the second layer, have a size of 3×3 . The total number of model parameters in Micronet was 182. Although this network is extremely small in comparison with state-of-the-art CNNs that have hundreds of millions of parameters, it is sufficient for illustrating the use of AI in SAS Viya for biomedical image analytics.

The entire set of 110 2-D, grayscale nodule patches (each of the 10 patients had 11 2-D patches) was randomly divided into two parts of approximately equal size, one for training Micronet, and the other for validating it. To prevent overfitting, the training set was expanded to about 750 images by using the `augmentImages` action, as follows:

```

s.image.augmentImages(
    imageTable = 'train',

```

```

cropList = [ {'useWholeImage': True,
             'mutations': {
                 'verticalFlip': True, 'horizontalFlip': True,
                 'sharpen': True, 'darken': True, 'lighten': True,
                 'colorJittering': True, 'colorShifting': True,
                 'rotateRight': True, 'rotateLeft': True,
                 'pyramidUp': True, 'pyramidDown': True } }],
casOut = vl(name='trainAug', replace=True))

```

Here, a set of new images was created from each image in the original training via operations such as flipping, rotation, and color changes. The output CAS table `trainAug` contains the original images along with the newly created ones.

Micronet was then trained asynchronously in 20 epochs as follows:

```

s.deepLearn.dltrain(
    model = 'microNet',
    table = 'trainAug',
    seed = 99,
    input = ['_image_', '_label_'],
    target = '_label_',
    nominal = ['_label_'],
    modelweights = vl(name='weights', replace=True),
    learningOpts = vl(miniBatchSize=1, maxEpochs=20, learningRate=0.001,
                      aSyncFreq=1, algorithm='ADAM'))

```

Note that the `_label_` column contained the true diagnosis for each image. The primary output of training is the optimal values of model parameters. These parameters are contained in the CAS table `weights`, which was then used to score against the validation set as follows:

```

s.dlscore(model = 'microNet',
           initWeights = 'weights',
           table = 'test',
           copyVars = ['_label_', "_image_"],
           layerOut = vl(name='layerOut', replace=True),
           casout = vl(name='scored', replace=True))

```

This scoring resulted in a misclassification error of about 5%. The error varies slightly between different executions of the solution because of the nondeterministic steps involved, including the random splitting of the data into two sets and the stochastic optimization in training.

DISCUSSION

The goals of this paper are to introduce the various SAS Viya components for biomedical image processing and to provide step-by-step illustrations of how to assemble those components to solve real-world biomedical image analytics problems. Two CAS action sets, `image` and `bioMedImage`, currently host all actions that directly operate on biomedical imagery. Lung nodule classification is used as an example to illustrate how to assemble these actions in combination with other SAS Viya actions to build pipelines that convert raw biomedical image data and annotations into insights that can help make decisions. Two biomedical image analytic approaches, one using machine learning (ML) and the other using artificial intelligence (AI) are demonstrated.

The choice between ML and AI is application-specific; both approaches have pros and cons. First, the ML solution to the lung nodule classification problem requires the segmentation of gray-scale images in order to separate the foreground (that is, the nodule pixels) from the background. Generally speaking, segmentation is a very challenging task and there is no single algorithm that works across all tissue types. In contrast, the AI solution operates directly on gray-scale images. Secondly, the ML solution provides a continuous metric, the degree of speculation (DOS). Such descriptive metrics are sometimes

helpful in clinical medicine, such as to assess progression of disease or response to therapy. The AI solution relies on optimization of CNN parameters that are based on data, and it provides only a binary classification of the images. By and large, it is not feasible to identify the physical meaning of various CNN parameters. Finally, the AI solution has a better classification accuracy than the ML solution, perhaps because the CNN parameters capture multiple shape features from the training data.

It is important to note that the methodologies and results in this paper are for illustrating SAS Viya capabilities; they are not clinically significant. In particular, more systematic studies with larger data sets have reported that shape features have less than 80% accuracy in classifying lung nodules (Niehaus, Raicu, Furst, and Armato 2015). The classification accuracies reported here are overestimated, because the example uses only 10 patients, who were selected from the TCIA data set based on image quality rather than selected randomly. Also, individual 2-D patches were treated as independent observations. In reality, 2-D slices from the same 3-D patch are correlated, and this dependence between slices leads to accuracy overestimation during cross validation.

CONCLUSION

With the recent release of SAS Viya, SAS has, for the first time, extended its platform to directly process and interpret biomedical image data. This new extension, available in SAS Visual Data Mining and Machine Learning, enables customers to load, visualize, process, and save health care image data and associated metadata at scale. Specific examples demonstrate how the new action sets, when combined with other data analytic capabilities available in SAS Viya, such as machine learning and artificial intelligence, empowers customers to assemble end-to-end solutions to significant, image-based health care problems. The complete source code of the examples demonstrated in this paper is publicly available free of cost (SAS Institute Inc. 2018).

Upcoming releases of SAS Viya will build on the foundation that this paper demonstrates. Future development efforts include elimination of the need to save intermediate results back to disk—for example by introducing the capability to process images with image-specific parameters. Also, the *bioMedImage* action set will be expanded by adding dedicated actions that perform standard segmentation and analysis of biomedical images.

REFERENCES

- Armato, S. G., Hadjiiski, L. M., Tourassi, G. D., Drukker, K., Giger, M. L., Li, F., Redmond, G., et al. 2015. "Special Section Guest Editorial: LUNGx Challenge for Computerized Lung Nodule Classification: Reflections and Lessons I Learned." *Journal of Medical Imaging*, 020103.
- Armato, S. G., Drukker, K., Li, F., Hadjiiski, L., Tourassi, G. D., Kirby, J. S., Clarke, L. P., et al. 2016. "LUNGx Challenge for Computerized Lung Nodule Classification." *Journal of Medical Imaging*, 044506.
- Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., et al. 2013. "The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository." *Journal of Digital Imaging*, 1045–1057.
- Johnson, H. J., McCormick, M. M., and Ibanez, L. 2017. *The ITK Software Guide: Introduction and Development Guidelines*. New York: Kitware, Inc.
- McWilliams, A., Tammemagi, M. C., Mayo, J. R., Roberts, H., Liu, G., Soghrati, K., Yasufuku, K., et al. 2013. "Probability of Cancer in Pulmonary Nodules Detected on First Screening CT." *New England Journal of Medicine*, 910–919.
- Niehaus, R., Raicu, D. S., Furst, J., and Armato, S. 2015. "Toward Understanding the Size Dependence of Shape Features for Predicting Spiculation in Lung Nodules for Computer-Aided Diagnosis." *Journal of Digital Imaging*, 704–717.
- Ramachandran, P., and Varoquaux, G. 2011. "Mayavi: 3D Visualization of Scientific Data" IEEE Computing in Science & Engineering. *Computing in Science & Engineering*, 40–51.
- SAS Institute, Inc. (2018, April 8). *SAS® ViyaTM Programming - Biomedical Image Analytics*. Retrieved from Github: <https://github.com/sassoftware/sas-viya-programming/blob/master/python/biomedical-image-analytics/lung-nodule-classification-sgf2018.ipynb>

ACKNOWLEDGMENTS

The authors thank the Society of Photographic Instrumentation Engineers (SPIE), American Association of Physicists in Medicine (AAPM), National Cancer Institute (NCI), and TCIA for providing all patient data that are used in this paper. They also thank Anne Baxter her for editorial review.

RECOMMENDED READING

- SAS® Visual Data Mining and Machine Learning 8.2: Programming Guide

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Fijoy Vadakkumpadan
SAS Institute, Inc.
919 531 1943
fijoy.vadakkumpadan@sas.com

Saratendu Sethi
SAS Institute, Inc.
919 531 0597
saratendu.sethi@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

How to Build a Recommendation Engine Using SAS® Viya®

Jared Dean, SAS Institute Inc., Cary, NC

ABSTRACT

Helping users find items of interest is useful and positive in nearly all situations. It increases employee productivity, product sales, customer loyalty, and so on. This capability is available and easy to use for SAS® Viya® customers. This paper describes each step of the process: 1) loading data into SAS Viya; 2) building a collaborative filtering recommendation model using factorization machines; 3) deploying the model for production use; and 4) integrating the model so that users can get on-demand results through a REST web service call. These steps are illustrated using the SAS Research and Development Library as an example. The library recommends titles to patrons using implicit feedback from their check-out history.

INTRODUCTION

Factorization machines are a common technique for creating user item recommendations, there is evidence they generate double digit increases in engagement and sales. SAS has had recommendation methods for many years including market basket analysis, K-nearest neighbors (KNN), and link analysis, along with other techniques for creating a next best offer. This paper focuses on creating recommendations using factorization machines and SAS® Viya® 3.3. The outcome of the paper is a recommendation engine that can be called from a RESTful API that returns the top five recommended books to library patrons. This process requires three main tasks be completed. (See Figure 1. Workflow for Recommendation Engine.) The tasks can be completed in any order, but all three must be completed before the service can be called through a RESTful service call.

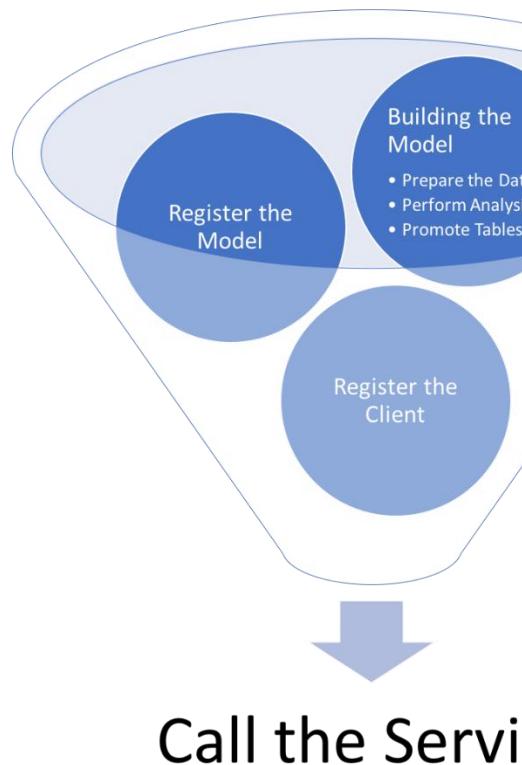


Figure 1. Workflow for Recommendation Engine

- Build the model. This is the analytical modeling section, which includes preparing the data, performing the factorization machine analysis, and creating the artifacts needed for providing recommendation requests on demand.
- Register the model. This is built in section two so that it is available to requestors on demand through a RESTful interface.
- Register the client within SAS Viya. This task is typically performed by a SAS administrator for the system, and the information is provided to the application developers.

The final section describes how the service can be called through a simple URL. This URL can then be embedded in an application, allowing SAS® Analytics to be part of your application in a simple and consistent manner.

GETTING STARTED

This paper uses SAS Viya 3.3, released in December 2017, to create a recommendation engine for the SAS R&D library. This application is meant to demonstrate the utility and ease of creating recommendations for your internal and external audiences using SAS Analytics. The technique used is a factorization machine. This example assumes that the FACTMAC is licensed.

Through the multiple language clients available for SAS Viya, several parts of this project can be accomplished using one of many programming languages. Examples are provided for you in SAS and Python. The Python code uses the SWAT package, which is available on GitHub at <https://github.com/sassoftware/python-swat>.

Three columns are required in the simplest application of a recommendation engine: User, Item, and Rating. More columns (attributes) can be used in creating recommendations, which is often referred to as tensor factorization. This factorization can add accuracy to your recommendations. All the columns used in factorization machine analysis must have values, and all the columns (except for the ratings) are treated as nominal variables.

The patron check-out data has various fields, but the fields that map to our application are the name of the patron (user) and the title (item) of the media the patron checked out. For an example, see Table 1. Example of Check-out History.

Table 1. Example of Check-out History

Name	Title
Dean Jared	Steve Jobs
Dean Jared	How Google works
Dean Jared	R for everyone advanced analytics and graphics
Dean Jared	Beautiful data the stories behind elegant data solutions
Dean Jared	Adapt why success always starts with failure
Dean Jared	Connectography mapping the future of global civilization
Dean Jared	Python in a nutshell
Dean Jared	Programming Python
Dean Jared	Practical statistics for data scientists 50 essential concepts

In the beginning of this project, I worked with a static copy of the data for development and validation. In production, the static copy of the data was replaced by a RESTful API call to get the latest library circulation data upon request.

Recommendations typically use a train/score model pattern. Here is the basic pattern: A model is trained on the most recent data available. After training is completed, the scoring tables are replaced with

updated versions. If the frequency for providing recommendations is very high (lots of users or users requesting recommendations often), you could have a continuous train/score cycle where as soon as the training ends it immediately gathers the latest data and begins the process again. For lower demand recommendation engines, you can schedule the training on a regular interval (hourly, daily, and so on). The time to train the model depends on the number of distinct user and item combinations (plus any additional attributes you include) and the number of transactions involved in the training. Factorization machines in SAS Viya can take advantage of parallel computing so that the elapsed time can be greatly reduced by using multiple CPUs.

BUILDING THE MODEL (TRAINING)

SETUP

The first step is to establish a connection to a CAS server. SAS® Cloud Analytic Services, the CAS server, is the next step for SAS in the evolution of SAS Analytics high-performance distributed processing on single or multiple machines.

Here is example SAS code:

```
options cashost="myserver.sas.com" casport=31004 casuser='Jared';
cas mysession;
```

Here is example Python code:

```
import swat
conn = swat.CAS('myserver.sas.com', 31004)
# Load the needed action sets
actionsets = ['astore', 'factmac', 'dataStep', 'fedSql']
[conn.builtins.loadactionset(i) for i in actionsets]
```

Notice the Python code has a few extra lines because the action sets must be loaded explicitly.

CREATE RATINGS

In a traditional recommendation setting, the items have ratings given by users (explicit feedback). In this example, ratings are not available, so a model is built using implicit feedback. For more information about creating implicit feedback, see the References and Recommended Reading sections.

To create quality recommendations without ratings, implicit feedback is used. Implicit feedback supplements our check-out history by randomly adding a book the user has not checked out for each book the user has checked out. This supplement creates a ratings data set that is twice the size of the actual check-out history. All of the books actually checked out by patrons receive a rating of 1, and all of the randomly selected books receive a rating of 0.

Here is a SAS macro, rate0, to generate implicit feedback:

```
%macro rate0(user);
proc sql;
  create table user as
  select distinct(title), (1) format=1. as rating,
    (&user.) as user
  from d.bhist
```

```

        where name = "&user.";
quit;

```

The preceding SQL procedure creates a distinct list of books for a specific user:

```

%let DSID = %sysfunc(open(user, IS));
%let n = %sysfunc(attrn(&DSID, NLOBS));
%let DSID=%sysfunc(close(&DSID));
proc sql outobs=&n.;
    create table rate0 as
        select title, (0) format=1. as rating, (&user.) as user
        from item
        except all
        select *
        from user
        order by ranuni(-1);
quit;

```

The preceding SQL procedure merges the user's books with all the books in the library, keeping only a random selection of the books the specific user did not check out and equal to the number they did check out.

```

proc append base=rate0_base data=rate0; run;
%mend rate0;

```

The remainder of the code partitions the check-out history and runs the rate0 macro for each user until there is a data set with all the actual check-out items that have a rating of 1 and all the randomly selected items that have a rating of 0. The data set has exactly twice as many records as the check-out history.

```

data item;
set d.bhist;
by title;
if first.title;
keep title;
run;
proc fedsql;
create table user_cnt as
select distinct(name) as "user"
from d.bhist
group by name
order by name;
quit;
filename file1 temp;
data _null_;
set user_cnt;
file file1;
put '%rate0(' name ')'; 
run;
proc delete data=rate0_base; run;
%include file1;

```

This macro takes a data set of the check-out history and returns a data set with the implicit feedback performed.

Here is a Python function to generate implicit feedback:

```

def sampleTitles(transhist: 'pd.DataFrame' = None,
                 user = 'name',
                 item = 'title') -> 'pd.DataFrame':
    nonco = pd.DataFrame()
    users = transhist[user].unique()
    for i in users:
        # get list of titles checked out
        titles = transhist.loc[transhist[user] == i]

        # get list of non-titles checked out
        nct = transhist.loc[~transhist[item].isin(titles[item].unique())]

        # randomly select non-checked out titles equal to the number of
        checkouts.
        samp = nct.sample(n=titles[item].count())[['bib', 'processed']]
        samp['rating'] = 0
        samp[user] = i

        nonco = nonco.append(samp)
    return nonco

```

The function takes a pandas dataframe, user, and item. The dataframe is of the borrowing history. User and name represent the columns in the dataframe that correspond to user and item. For this example, the patron is the user, and the book is the item.

Regardless of the programming language (SAS, Python, R, and so on), here is the procedure for generating implicit feedback:

1. Create a unique list of all the patrons.
2. Create a unique list of the books each patron has checked out and the total number of checkouts. A random book is selected each time a book is checked out.
3. Create a list of all titles offered by the library. If there are multiple copies or media (audiobook, e-book, hardback, and so on), they are treated as a single title.
4. Sample without replacement from the universe of titles that the user has not checked out. This is represented by the blue area in **Error! Reference source not found..**

The circle represents all the titles. The white area is books the patron has checked out. The shaded area is books that have not been checked out by the patron.

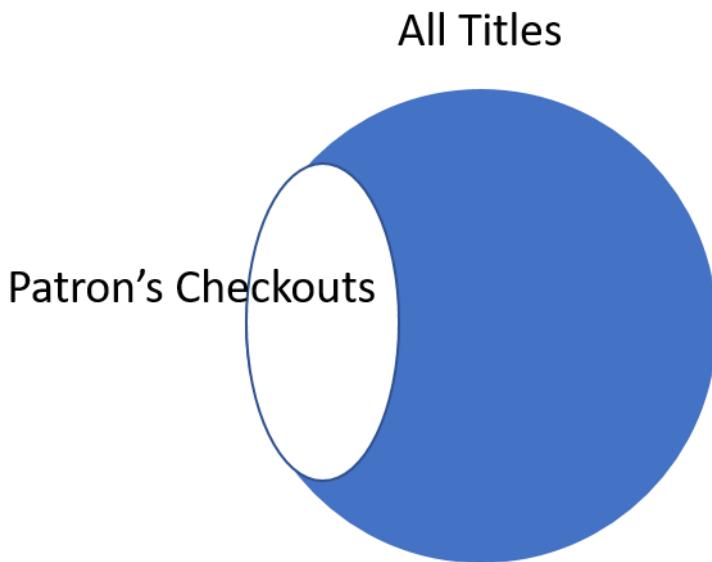


Figure 2. Illustration of Sampling Design

5. Add the sampled books to the check-out history.

This procedure is then repeated for each patron in the library. For more information about implicit feedback, see the References and Recommended Reading sections.

With the implicit feedback completed, a sample of our data now looks like Table 2. Example Data after Implicit Feedback. The books with rating 1 are books I have checked out from the SAS library. The books with rating 0, I have not checked out. The complete table would include the check-out history for each library patron. The books checked out by each patron have a rating of 1, and all the randomly selected books that were not checked out have a rating of 0.

Table 2. Example Data after Implicit Feedback

Name	Title	Rating
Dean Jared	Steve Jobs	1
Dean Jared	How Google works	1
Dean Jared	R for everyone advanced analytics and graphics	1
	Beautiful data the stories behind elegant data	
Dean Jared	solutions	1
	Connectography mapping the future of global	
Dean Jared	civilization	1
Dean Jared	Adapt why success always starts with failure	1
Dean Jared	Python in a nutshell	1
Dean Jared	Programming Python	1
	Practical statistics for data scientists 50 essential	
Dean Jared	concepts	1
Dean Jared	HTML5 up and running	0
Dean Jared	Wordpress for dummies	0
Dean Jared	PHP and MySQL by example	0
Dean Jared	Adobe Photoshop CS5 classroom in a book	0

Dean Jared	Exploratory factor analysis	0
Dean Jared	Spatial statistics	0
	SAS certification prep guide base programming for	
Dean Jared	SAS 9	0
Dean Jared	Beginning Lua programming	0
Dean Jared	Head First Excel	0

CREATE RECOMMENDATIONS

Now that we have a variety of ratings in the data, the data can be loaded into CAS and a factorization machine analysis performed.

Here is the SAS code to load the data and run the FACTMAC procedure:

```
libname mycas cas;
data mycas.checkout;
  set final_rating;
run;

proc factmac data=mycas.checkout outmodel=mycas.factors_out;
  autotune;
  input Name Title /level=nominal;
  target rating /level=interval;
  savestate;
  output out=mycas.score_out1 copyvars=(rating);
run;
```

Here is the Python code to load the data and run the FACTMAC action:

```
conn.upload(casout={'name':'checkout', 'replace':True},
            data=final_rating.dropna())
rec1 = conn.factmac(table='checkout',
                     inputs      = ['Name', 'Title'],
                     nominals   = ['Name', 'Title'],
                     id         = ['Name', 'Title'],
                     target     = 'rating',
                     nfactors   = 10,
                     maxiter    = 100,
                     learnstep= 0.15,
                     seed=9878,
                     output=  {'casout':{'name':'score_out1',
                                         'replace':'TRUE'},
                               'copyvars':['rating']},
                     outModel={'name':'factors_out', 'replace':'TRUE'},
                     saveState={'name':'state'},
)

```

Regardless of which interface we use to run the analysis, there are several details that need to be specified.

The INPUTS, ID, and TARGET statements must be specified. The number of factors (nfactors), maximum iterations (maxiter), and the learning rate (learnstep) variables have defaults but can be specified by the user or optimal settings can be found using autotuning. I have explicitly listed the options here for clarity. The quality of a factorization machine is based on the root mean squared error (RMSE). For more information about the FACTMAC syntax, see the Recommended Reading section.

To facilitate making recommendations (scoring) on demand for users, we need to save the model in an ASTORE object. An ASTORE is a compressed binary representation of the model. Saving the model is accomplished in the OUTMODEL statement. For more information about ASTORE, see the Recommended Reading section.

PROMOTE TABLES

By default, CAS tables are available only in the session that created them. To make them available globally for requests on demand, we must promote the tables.

Three tables must be promoted for this application:

1. the ASTORE from the SAVESTATE statement. This is used for recommending books to returning patrons.
2. the factors table from the OUTMODEL statement. This is used for recommending books to new patrons.
3. the borrower history from the DATA statement. This is used for creating a list of distinct books at the time a recommendation is requested.

Here is the SAS code to promote the needed tables:

```
proc casutil;
  promote casdata="checkout" casout='libraryrec_latest';
  promote casdata="state" casout='libAstore_latest';
  promote casdata="factors_out" casout='factors_latest';
quit;
```

Here is the Python code to promote the needed tables:

```
conn.droptable(name='libraryrec_latest', quiet=True)
conn.droptable(name='libAstore_latest', quiet=True)
conn.droptable(name='factors_latest', quiet=True)
conn.promote(name='checkout', target='libraryrec_latest')
conn.promote(name='state', target='libAstore_latest')

conn.promote(name='factors_out', target='factors_latest')
```

With these tables promoted, the training portion is complete. The next sections demonstrate how to register the model as a service in SAS Viya, how to register the client so that it can be called as a service, and how to make a RESTful API call to provide recommendations on demand.

REGISTERING THE MODEL

The SAS Viya infrastructure has many micro services. For this application, I used the SAS Job Execution service because I found it the simplest to work with. There is a user interface specifically designed to help you register SAS jobs, which is experimental in SAS Viya 3.3 (released in December 2017).

Your SAS administrator should provide you with a URL. For this paper, assume it is <http://myviya.sas.com>.

When you open that link in your browser, you are prompted to sign in.



Figure 3. Sign-in Screen

After a successful sign-in, you will likely be redirected to <http://myviya.sas.com/SASHome> and your dashboard will look like Figure 4. SASHome Dashboard.

The image shows the SASHome Dashboard. At the top, there's a banner with icons for "SAS Home", "Search", and user information. Below the banner, a "Welcome!" message is displayed. There are several interactive tiles: "Explore and Visualize Data" (orange), "Build Models" (green), "Manage Environment" (blue), "Prepare Data" (yellow), and "Manage Themes" (purple). Below these are three sections: "Recent" (with an "Open" button), "Favorites" (with an "Add Favorites" button), and "Links" (with an "Add Links" button). A navigation bar at the bottom includes "Browse", "Shortcut", "Tile", "Publish", and a "Hide" button.

Figure 4. SASHome Dashboard

Next, navigate to <http://myviya.sas.com/SASJobExecution/admin>.

Some important items to note:

- The first part of the URL will be different for your organization.
- The URL is case sensitive.
- You must have administrative rights in SAS Viya to register a job with the SAS Job Execution service.

Your browser should now display the SAS Job Execution client. (See Figure 5. SAS Job Execution Client.)

The screenshot shows the SAS® Job Execution (Experimental) interface. At the top is a dark blue header bar with the title "SAS® Job Execution (Experimental)". Below the header is a toolbar with several icons: a magnifying glass, a trash can, a gear, a plus sign, a file folder, a document, a list, a search icon, a camera, a refresh, and three vertical dots. To the right of the toolbar is a vertical sidebar titled "Folders". The sidebar lists several categories with corresponding icons: "My Folder" (file folder), "Model Repositories" (file folder), "Products" (file folder), "Projects" (file folder), "Public" (file folder), and "Users" (file folder). Each category has a circular arrow icon to its right. At the bottom of the sidebar is a small upward-pointing arrow icon.

Figure 5. SAS Job Execution Client

This step (Job Execution service) does not currently support code from other languages such as R or Python. Therefore, you need to write exclusively SAS code.

You need to decide where to store your SAS code. From the toolbar at the top of the screen, you can navigate the folders and create new folders and programs.

Here is the SAS code that runs each time the RESTful API is called. Each code block is explained following the code. I have left commented parts of the code so that if your application has different requirements, you can use it as a template.

```
/* Close all ods destinations */
ods _all_ close;
/* for debug - print all the macro variables */
*%put _global_;
/* To create HTML output */
*filename _webout sasfsvam parenturi="&SYS_JES_JOB_URI" name='__webout.htm';
*ods html5 file=_webout style=HTMLBlue;

filename _webout sasfsvam parenturi="&SYS_JES_JOB_URI" name='__webout.json';
```

The preceding code closes all the output destinations and establishes a filename that will be in a JSON file to return to the requestor.

```
options cashost="ip.or.url.com" casport=<> ;
/* establish a CAS session */
cas mysession;
```

```
/* create a libname to your CAS session */
libname mycas cas;
```

The preceding block of code creates the CAS session and creates a library reference between the SAS session and the CAS server. You need to specify these items:

- cashost (using IP address, DNS name, or localhost)
- casport (provided by your administrator)

Note: If you are authenticating using OAuth do not specify the casuser in the options this will over ride the OAuth authentication.

Next, we use several statements within the CAS procedure to prepare the data, create ratings, and determine which books to recommend.

These are the action sets that are needed:

```
proc cas;
    loadactionset "dataStep";
    loadactionset "fedSql";
    loadactionset " astore";
run;
```

The ASTORE object we produced earlier in the Promote Tables section of the paper takes a table with patrons and titles and returns a predicted rating. In this DATA step code, we need to prepare a data set for scoring. We create two columns—one of the user, and one for each title in the library collection. The variable bib is an identifier for the title of the book.

```
/* Drop and rename */
dataStep.runCode code =
  data user_rec;
    set libraryrec_latest;
    by bib;
    if first.bib;
    empno= lowercase("'"&score_user"'");
    keep empno bib processed;
  run;";
run;
```

In the SCORE statement, we pass the CAS table we just created and create an output table named ranked_books.

```
/* Score with Astore */
astore.score /
  table = 'user_rec'
  rstore='libAstore_latest'
  out = {name='ranked_books' replace=True}
  ;
run;
```

In the following SUMMARY action, we find the max rating. The FACTMAC ASTORE returns a rating of missing for any row where the patron or book is missing. A missing value is less than any other number in SAS, so if the max is missing that means all the values are missing.

```

/* Find max rating. If all ratings are missing then new user. */
simple.summary result=m /
  table='ranked_books'
  subset={'max'}
  inputs={'P_rating'};

/* drop the table in preparation to replace it */
table.droptable /
  name='book_recs'
  quiet=True;

```

In the following block of code, we check the max value from the ranked_books table. If the max value is missing, it means this is a new user. We do not have any history with new users, so we will recommend the most popular books in the library. In both cases, we create a table named book_recs with the top five recommendations.

```

if missing(m.summary[1,2]) then do;
  fedsqlexecdirect result=top5rec /
    query="create table book_recs as
      select a.Level as bib, b.processed
      from factors_latest a, user_rec b
      where Variable='bib' and a.level=b.bib
      order by Bias desc limit 5;";
end;
else do;
  fedsqlexecdirect result=top5rec /
    query="create table book_recs as
      select bib, processed
      from ranked_books
      where P_rating^=.
      order by P_rating
      desc limit 5;";
end;
run;
quit;

```

In the following block of code, the recommendation table is written as JSON output, which is returned to the application that called the RESTful API. JSON is the standard return format for REST API calls.

```

proc json out=_webout;
  export mycas.book_recs(keep=title);
run;

/* code to use for HTML results or debug */
/*
proc print data=mycas.ranked_books(obs=5);
run;
proc print data=mycas.book_recs;
run;
ods html5 close;
*/

```

REGISTERING THE CLIENT

Before we can call our recommendation scoring service, we must register the client with SAS Logon. This task is typically performed by the SAS administrator, not the application developer, but it must be completed before anything will work. For more information, see “Obtain an ID Token to Register a New Client ID” in the References section. Registering the client is needed to ensure that the application is authorized. The process involves generating a token as an authorized user, and then using that token to authorize this application.

The referenced documentation goes into more detail, but here is the high-level process:

1. Get a consul token from the system files:
 - a. cd /opt/sas/viya/config/etc/SASSecurityCertificateFramework/tokens/consul/default
 - b. sudo export CONSUL_TOKEN=`cat client.token`
2. Get a client access token to register the client:
 - a. curl -X POST "http://localhost/SASLogon/oauth/clients/consul?callback=false&serviceId=horizonapp" -H "X-Consul-Token: <*****-****-****-*****>"
3. Save the client access token for future commands:
 - a. export TOKEN=eyJhbGc...PDKgg
4. Register the new client. Give it a name and assign it a secret password:
 - a. curl -X POST "http://localhost/SASLogon/oauth/clients" -H "Content-Type: application/json" -H "Authorization: Bearer \$TOKEN" -d '{"client_id": "mysuperapp", "client_secret": "<SECRET_PASSWORD>", "scope": ["openid", "openstackusers"], "authorized_grant_types": ["client_credentials"]}'

Note: In step 4, use “client_credentials” as the authorized grant type instead of a password for improved security.

CALLING THE SERVICE (SCORING)

Because of the work we did to register the model, calling the service is very simple. An authorized user can make a simple REST call to the SASJobExecution endpoint. There are two ways to call programs that are registered for the SAS Job Execution service. You can reference the program by the job definition ID as shown here:

http://myviya.sas.com/SASJobExecution/?_job=/jobDefinitions/definitions/1404f786-2358-48bb-a41f-f82b2a6a0791&score_user='John Doe'

Or, you can use the path to the program as shown here:

http://myviya.sas.com/SASJobExecution/?_program=/Public/libraryRecScore &score_user='Jane Doe'

Both calls yield the same results. It is personal preference which one you would like to call. After completing the steps in this paper, you should be able to paste a URL similar to either preceding call and get JSON results displayed in your browser.

When either call is made, JSON is returned. The JSON response is not intended to be read by users, but it will be processed. Here is an example of the code that is returned:

```
{"SASJSONExport": "1.0", "SASTableData+BOOK_RECS": [{"BIB": "77949", "processed": "High performance habits how extraordinary people become that way"}, {"BIB": "7860", "processed": "Proceedings of the fifth annual SAS Users Group International SUGI Conference San Antonio Texas February 18 20"}]
```

```
1980"}, {"BIB": "8159", "processed": "SEUGI 93 proceedings of the eleventh SAS European Users Group International Conference Jersey U K June 22 25 1993"}, {"BIB": "7873", "processed": "Proceedings of the tenth annual SAS users group international conference SUGI 10"}, {"BIB": "9399", "processed": "Step by step programming with base SAS software"}]]}
```

As a SAS programmer, you might not have any experience calling a RESTful service and using the JSON response, but the web developers in your organization use these tools all the time. You can now quickly and efficiently provide easy access to SAS Analytics in the applications that your organization is building.

CONCLUSION

Factorization machines are a modern recommendation technique using SAS Viya 3.3 that you can easily incorporate in your applications to give users suggestions and guidance. To create a recommendation engine takes four steps: training a model, registering the model, registering the client, and calling the RESTful service.

To train the model, you gather the data, create ratings if they do not already exist, perform a factorization machine analysis, and finally save the results to create on-demand recommendations.

Registering the model must be written in SAS code, and the user must have administrator rights in SAS Viya. This is the code that runs each time the API is called.

Registering the client is usually done by a SAS administrator.

Calling the RESTful service makes it simple to embed SAS Analytics in your application with JSON results being returned.

By following these steps, you can unleash the power of SAS in your applications in a straightforward way.

REFERENCES

Henry, Joseph. "Show Off Your OAuth." *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available
<http://support.sas.com/resources/papers/proceedings17/SAS0224-2017.pdf>.

SAS Documentation 2017. "Obtain an ID Token to Register a New Client ID." *Encryption in SAS Viya 3.2: Data in Motion*. Accessed December 4, 2017.
<http://go.documentation.sas.com/?cdId=calcdc&cdcVersion=3.2&docsetId=secref&docsetTarget=n1xdqv1sezryahn17erzcunxwix9.htm&locale=en#p1w1tzdisw4147n1hlc8cwhi9fwg>

SAS Documentation 2017. "Obtain an Access Token Using Password Credentials." *Encryption in SAS Viya 3.2: Data in Motion*. Accessed December 21, 2017.
<http://go.documentation.sas.com/?cdId=calcdc&cdcVersion=3.3&docsetId=calauthmdl&docsetTarget=n1pkgyrtk8bp4zn1d0v1ln4869og.htm&locale=en#p0lxoq5bx2i6t8n13b3y3tcjwj9v>

Hu, Y., Y. Koren, and C. Volinsky. 2008. "Collaborative Filtering for Implicit Feedback Datasets." *Proceedings of the Eighth IEEE International Conference on Data Mining*. Pisa, Italy. pp. 263-272. doi: 10.1109/ICDM.2008.22

Grau, J., [Personalized product recommendations: predicting shoppers' needs](#). eMarketer, March 2009

Rendle, Steffen. 2010 "Factorization machines." *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE.

SAS Documentation 2017. "SAS Cloud Analytic Services (CAS)." *Differences in the SAS 9 and SAS Viya 3.2 Platforms*. Accessed January 1, 2018.

<http://go.documentation.sas.com/?docsetId=whatsdiff&docsetTarget=p1gfaswb875orbn1xn4ao8b8jbfq.htm&docsetVersion=3.2&locale=en>

RECOMMENDED READING

- SAS Institute Inc. 2016. "The ASTORE Procedure." *SAS Visual Data Mining and Machine Learning 8.1: Data Mining and Machine Learning Procedures*. Cary, NC: SAS Institute Inc. Available http://go.documentation.sas.com/?docsetId=casm1&docsetTarget=viyaml_astore_toc.htm&docsetVersion=8.1&locale=en
- SAS Institute Inc. 2016. "The FACTMAC Procedure." *SAS Visual Data Mining and Machine Learning 8.1: Data Mining and Machine Learning Procedures*. Cary, NC: SAS Institute Inc. Available http://go.documentation.sas.com/?docsetId=casm1&docsetTarget=viyaml_factmac_toc.htm&docsetVersion=8.1&locale=en
- Hu, Y., Y. Koren, and C. Volinsky. "Collaborative Filtering for Implicit Feedback Datasets." 2008. *Proceedings of the Eighth IEEE International Conference on Data Mining*. Pisa, Italy. 2008. pp. 263-272. doi: 10.1109/ICDM.2008.22
- Silva, Jorge and Wright, Raymond E. "Factorization Machines: A New Tool for Sparse Data" *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available <http://support.sas.com/resources/papers/proceedings17/SAS0388-2017.pdf>

ACKNOWLEDGMENTS

The author would like to thank Jorge Silva, Joseph Henry, Sath Sourisak, Mike Roda, Vince DelGobbo, Matt Bailey, and Brett Wujek for their contributions to the paper. He is also grateful to Tate Renner for her editorial contributions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Jared Dean
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Jared.Dean@sas.com
<http://www.sas.com>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Ready to take your SAS® and JMP® skills up a notch?



Be among the first to know about new books,
special events, and exclusive discounts.

support.sas.com/newbooks

Share your expertise. Write a book with SAS.

support.sas.com/publish

 sas.com/books
for additional books and resources.


THE POWER TO KNOW®